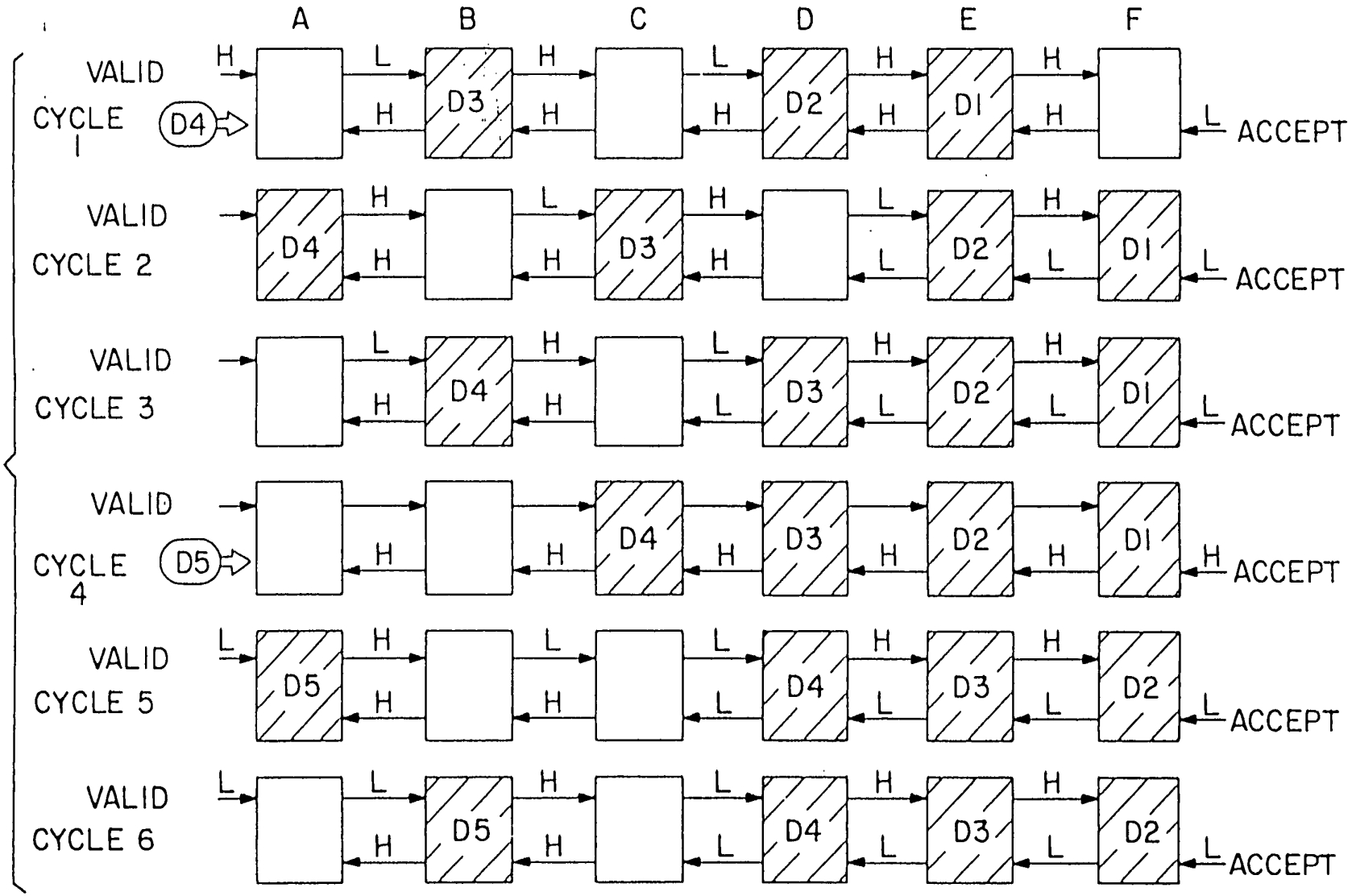

A
B
C
D
E
F
VALID
CYCLE 1
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 2
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 3
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 4
D5
D4
D3
D2
D1
ACCEPT
VALID
CYCLE 5
D5
D4
D3
D2
ACCEPT
VALID
CYCLE 6
D5
D4
D3
D2
ACCEPT
H
L

FIG. 1

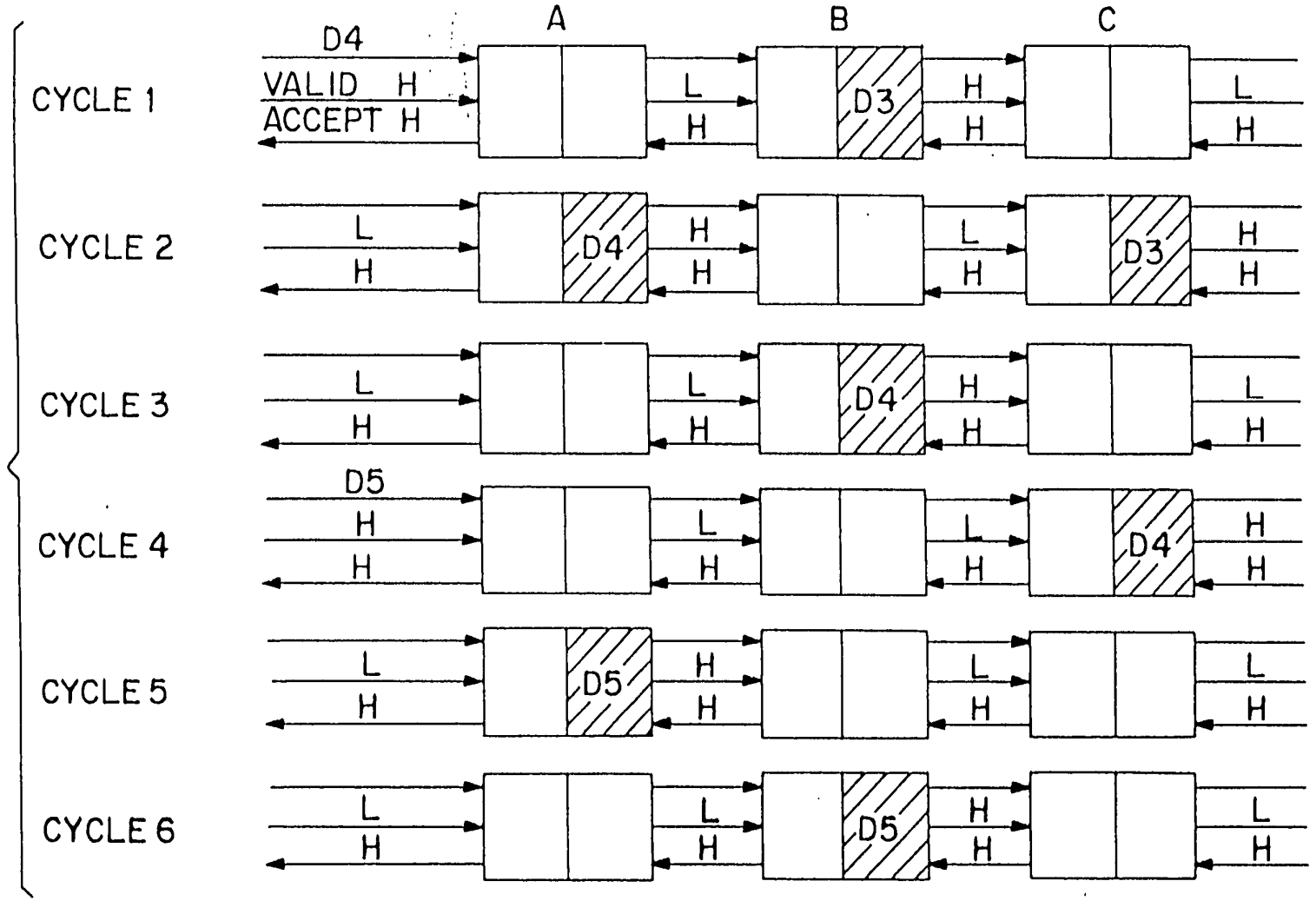

A
B
C
CYCLE 1
D4
VALID H
ACCEPT H
L
H
D3
H
H
L
H
CYCLE 2
L
H
D4
H
H
L
H
D3
H
H
CYCLE 3
L
H
L
H
D4
H
H
L
H
CYCLE 4
D5
H
H
L
H
L
H
D4
H
H
CYCLE 5
L
H
D5
H
H
L
H
L
H
CYCLE 6
L
H
L
H
D5
H
H
L
H

FIG. 2(A)

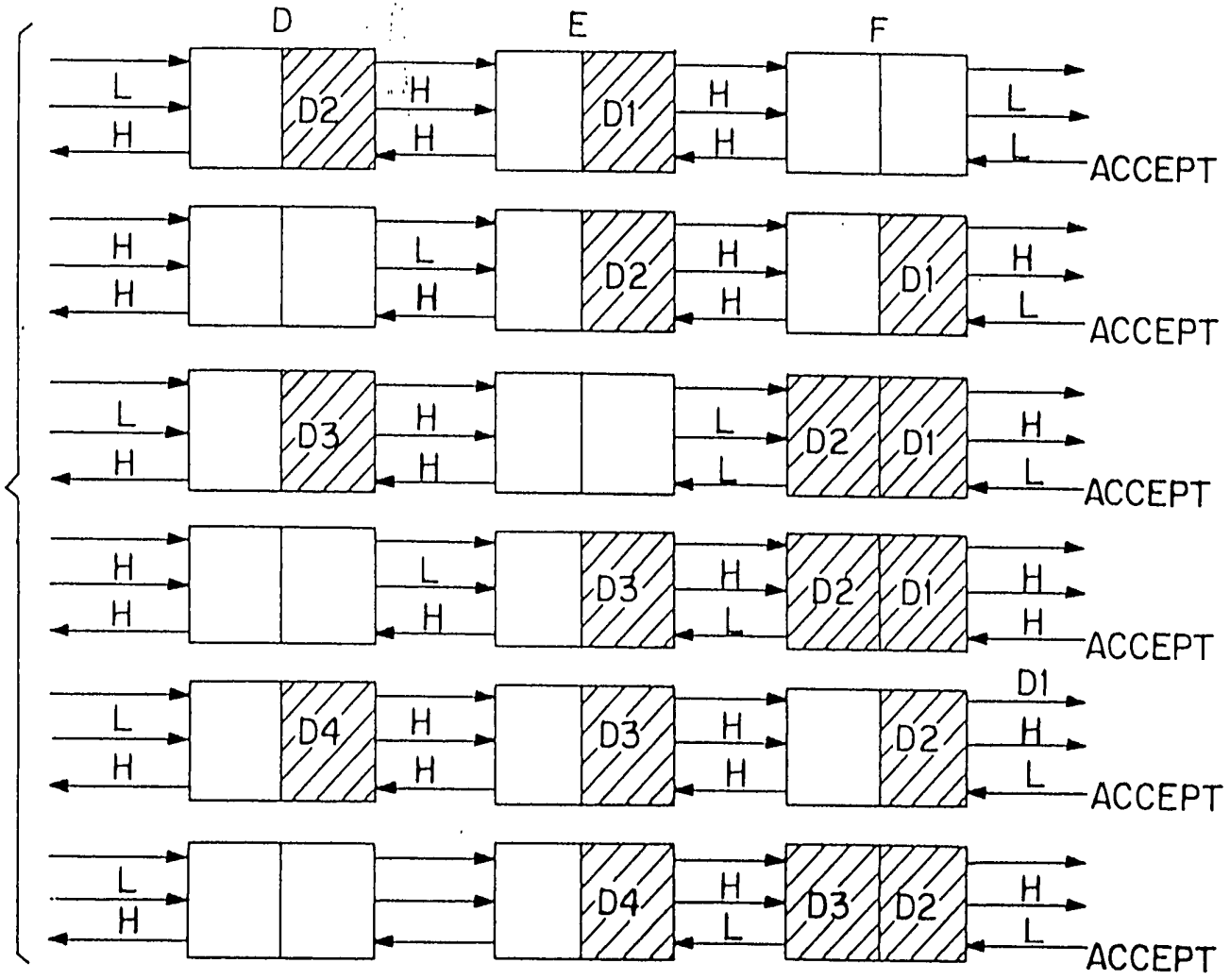


FIG. 2(B)

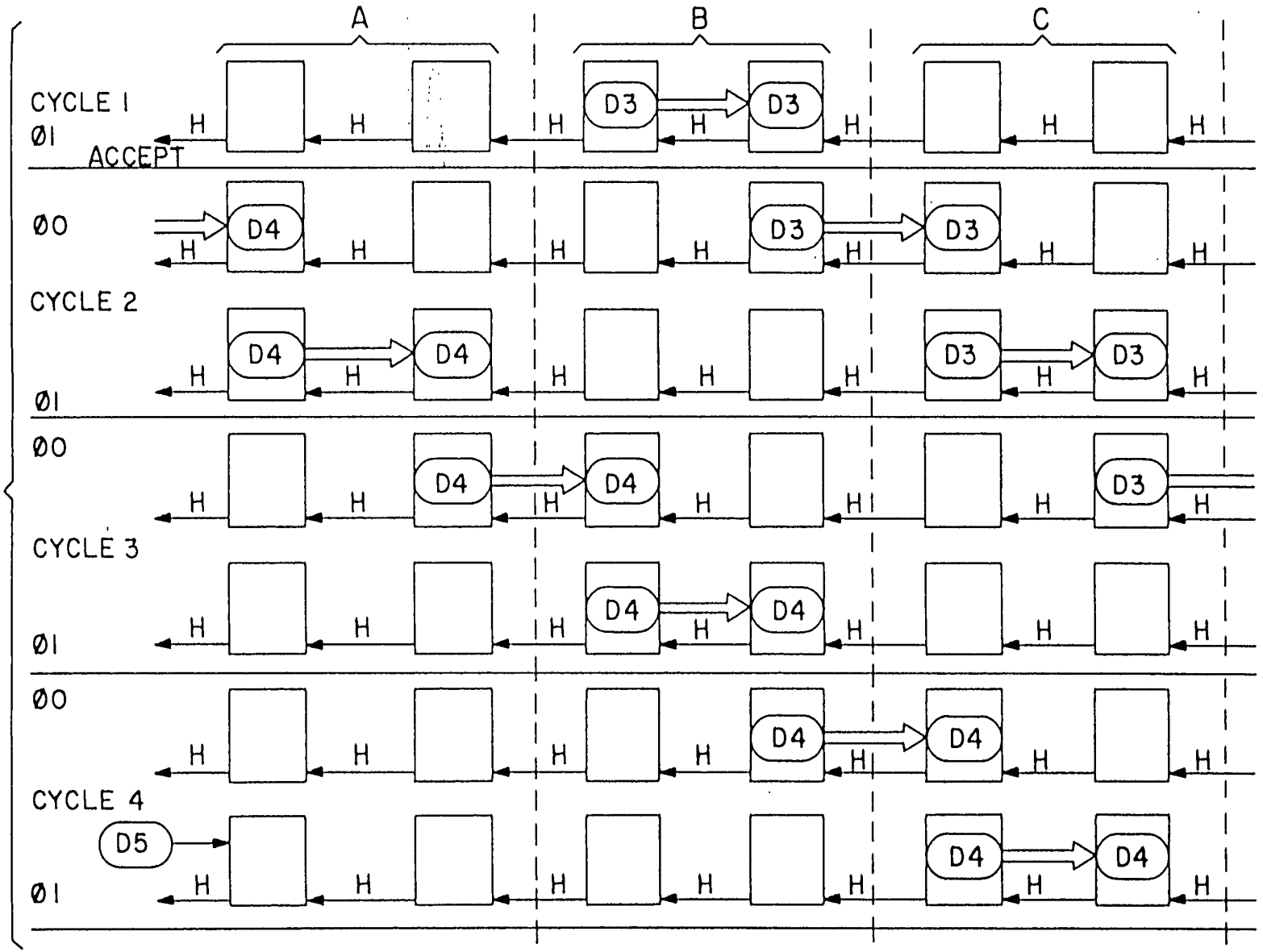

A
B
C
CYCLE 1
Ø1
ACCEPT
Ø0
CYCLE 2
Ø1
Ø0
CYCLE 3
Ø1
Ø0
CYCLE 4
Ø1
D3
D4
D5
H

FIG. 3A-1

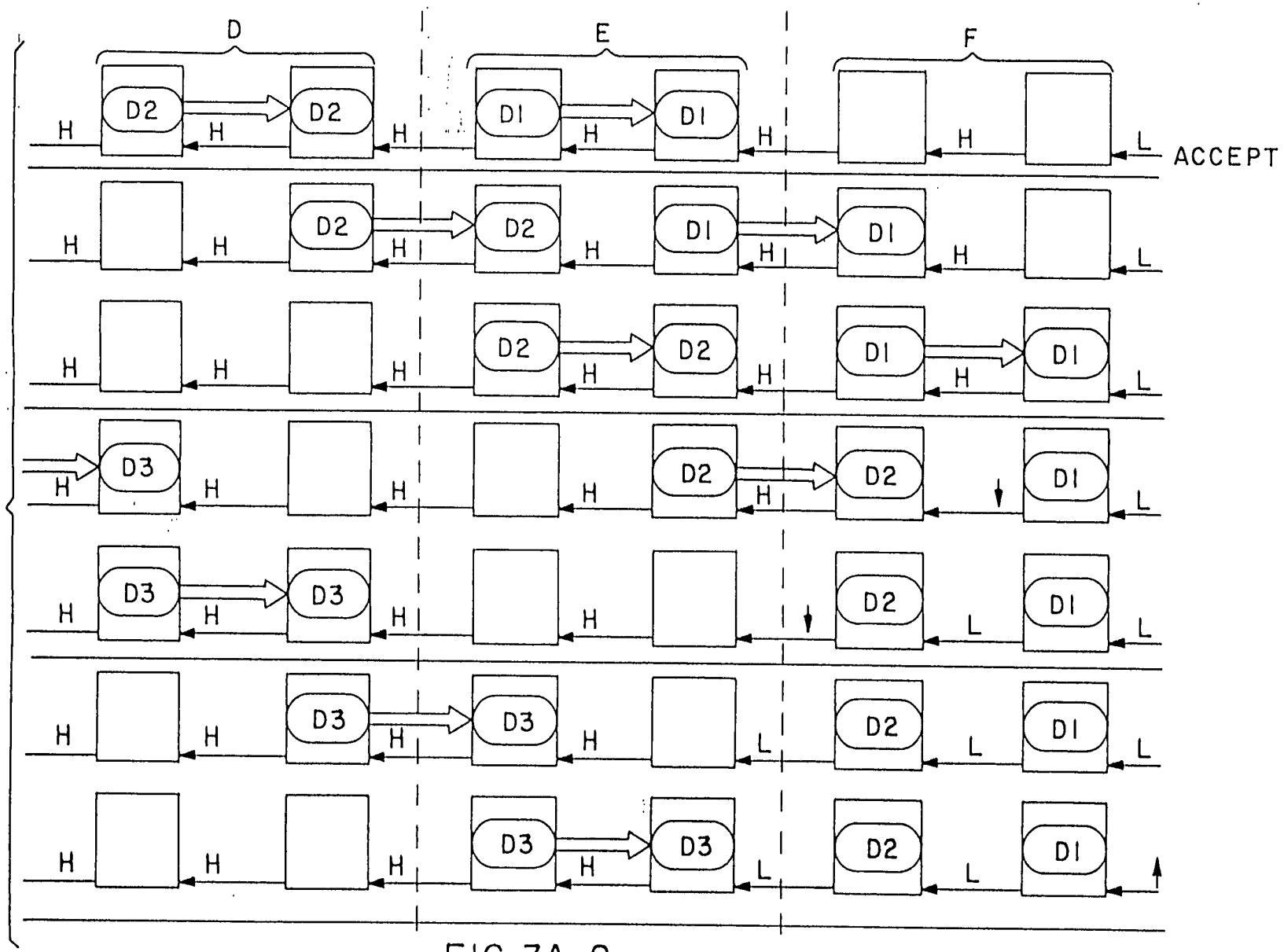


FIG. 3A-2

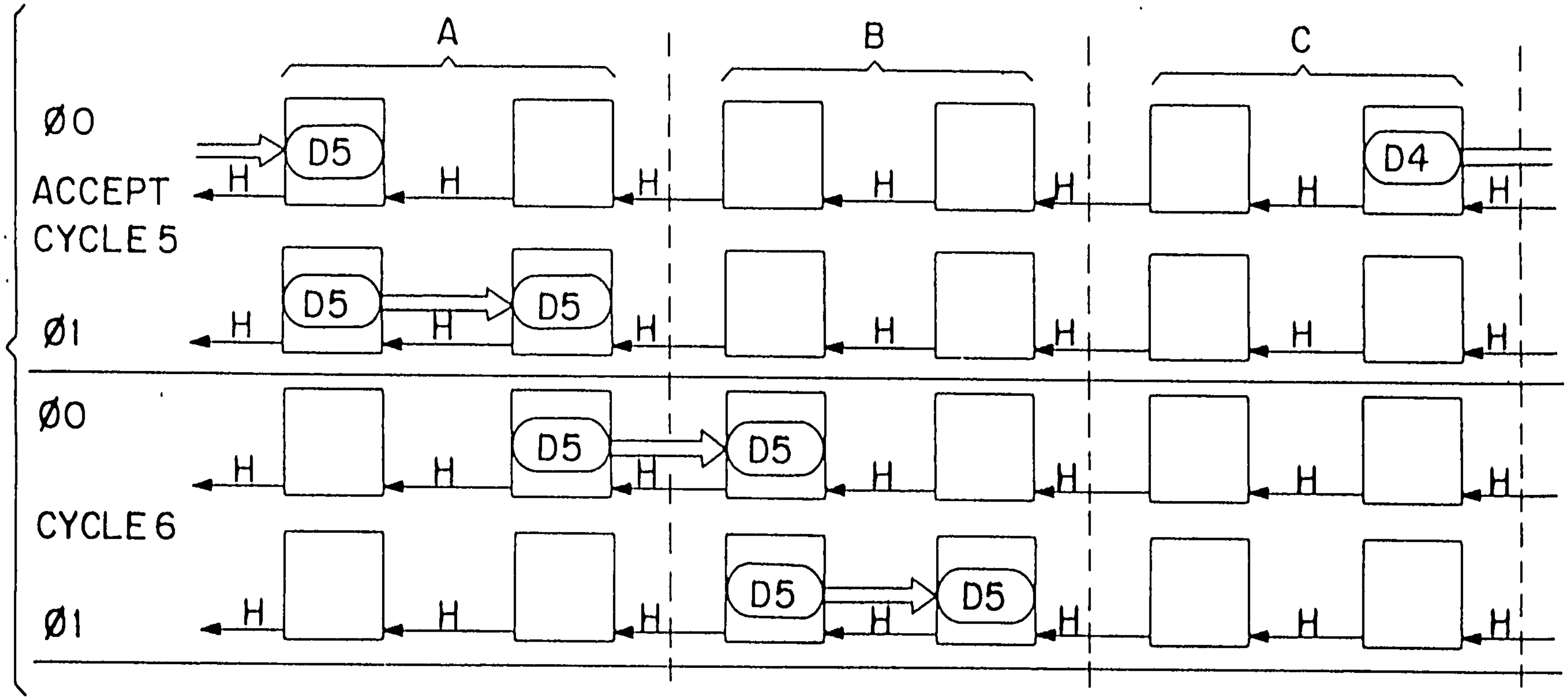


FIG. 3B-1

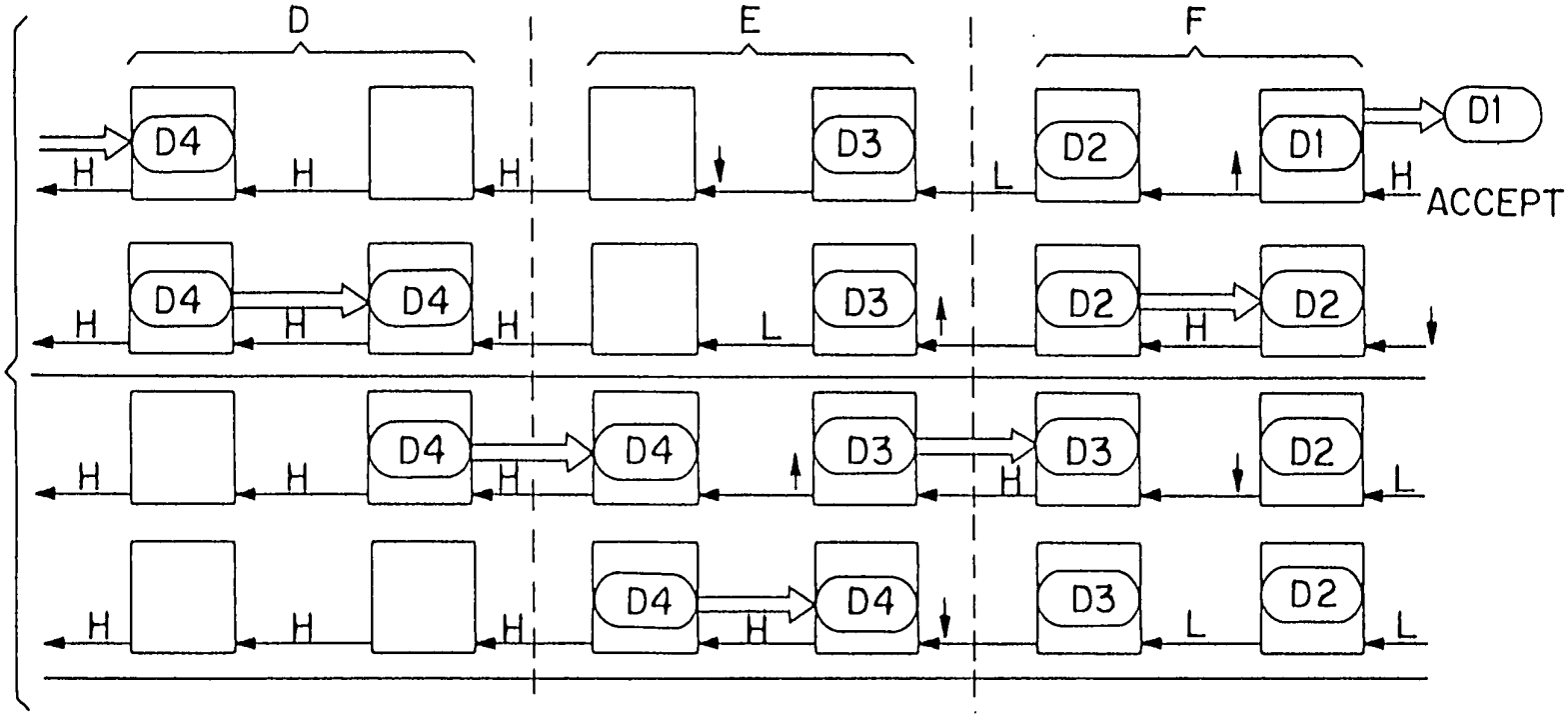


FIG. 3B-2

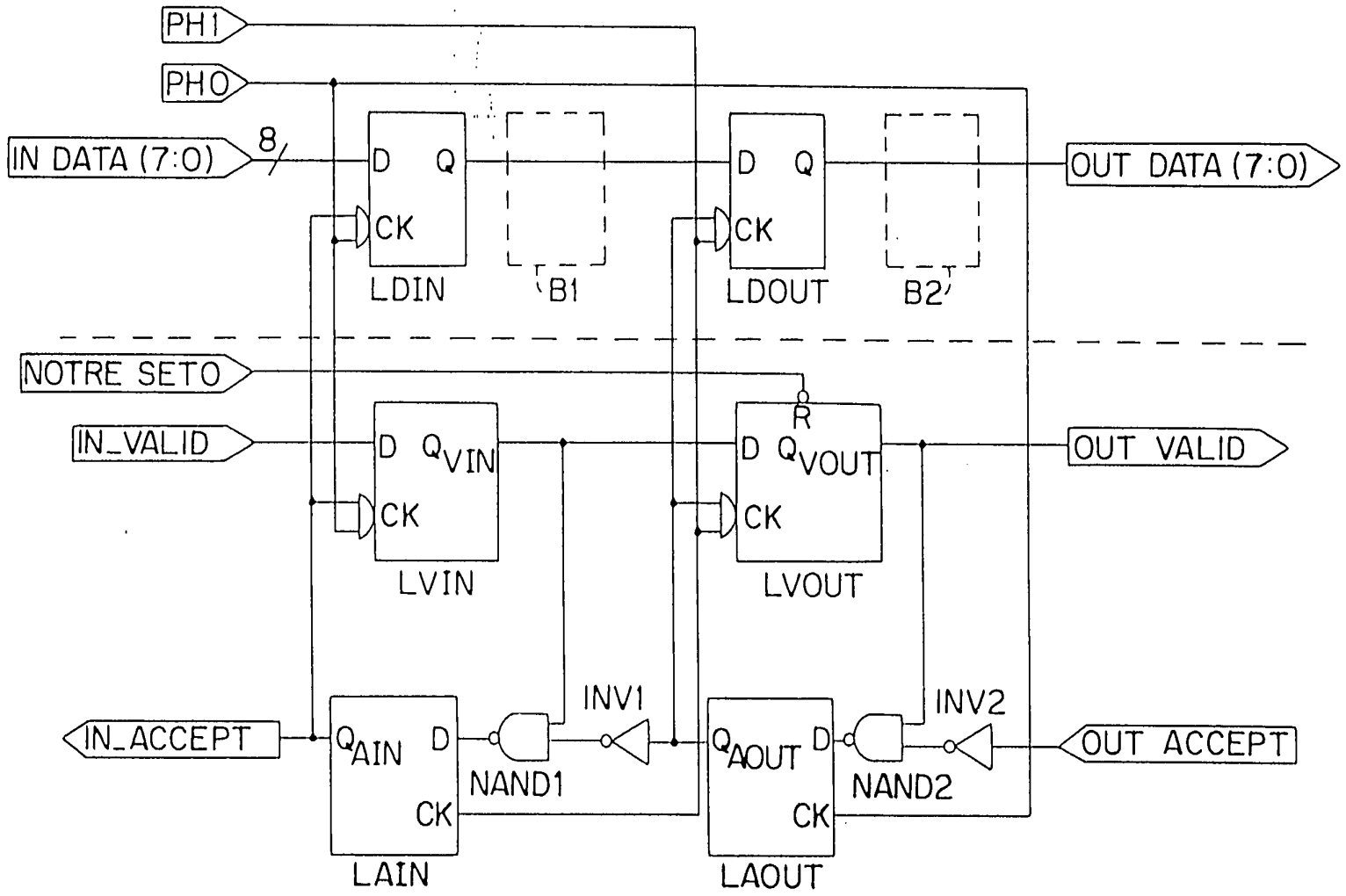

PH1
PH0
IN DATA (7:0)
8
D Q
CK
LDIN
B1
D Q
CK
LDOUT
B2
OUT DATA (7:0)
NOTRE SETO
R
IN_VALID
D Q VIN
CK
LVIN
D Q VOUT
CK
LVOUT
OUT VALID
IN_ACCEPT
Q AIN D
CK
LAIN
NAND1
INV1
Q AOUT D
CK
LAOUT
NAND2
INV2
OUT ACCEPT

FIG. 4

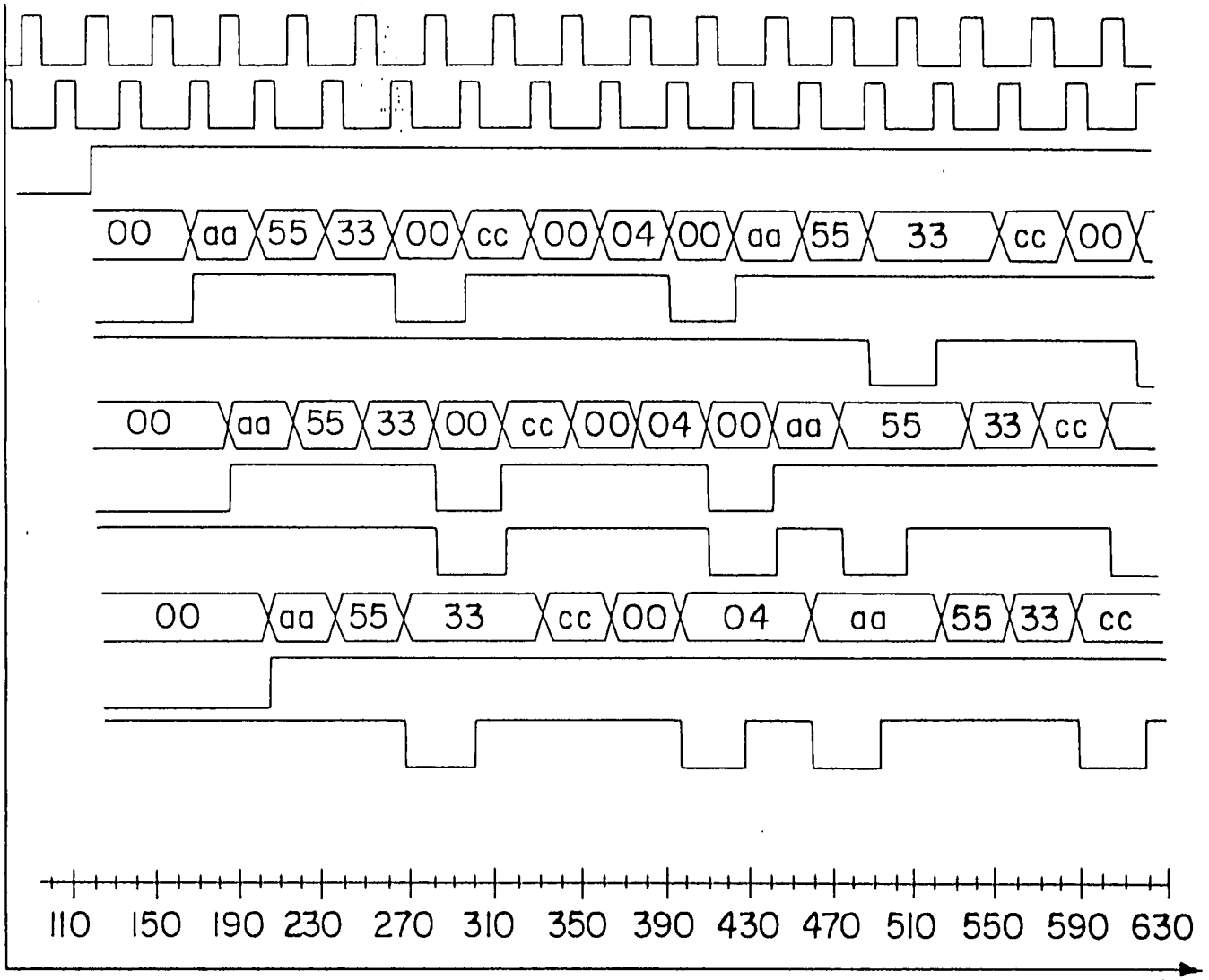

00 aa 55 33 00 cc 00 04 00 aa 55 33 cc 00
00 aa 55 33 00 cc 00 04 00 aa 55 33 cc
00 aa 55 33 cc 00 04 aa 55 33 cc
110 150 190 230 270 310 350 390 430 470 510 550 590 630
TIME

FIG. 5(A)

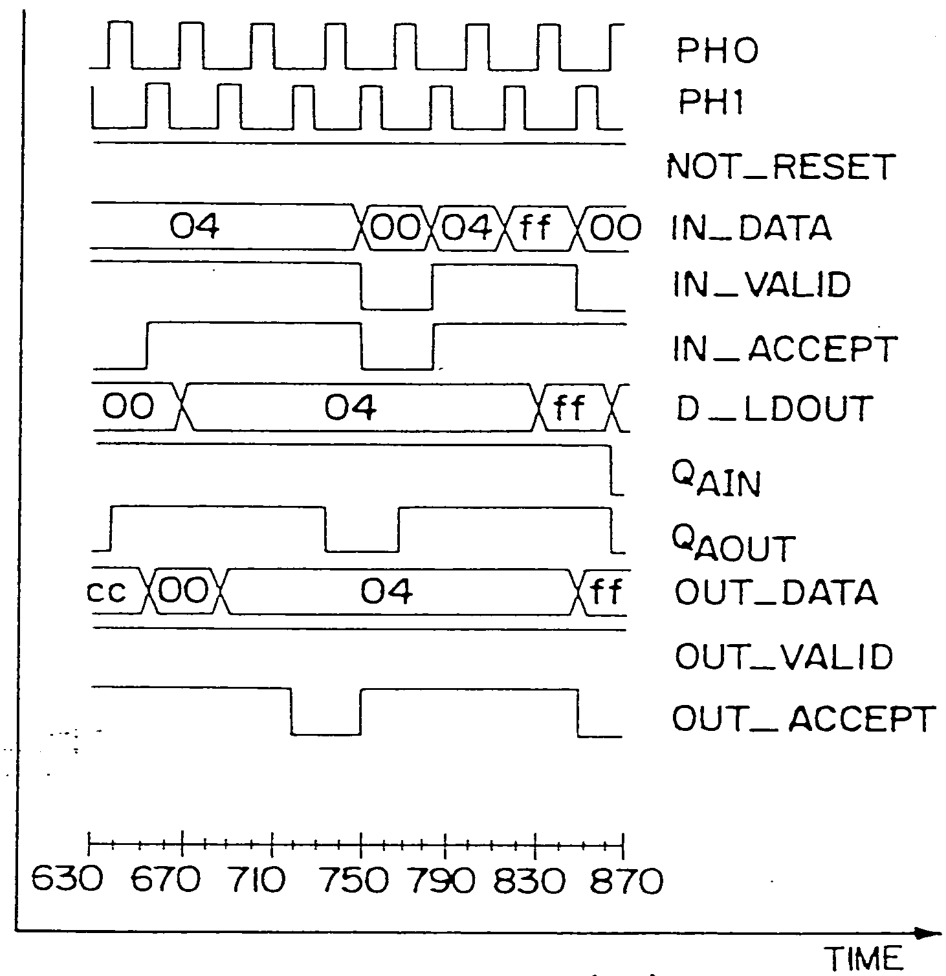

PHO
PH1
NOT_RESET
IN_DATA
IN_VALID
IN_ACCEPT
D_LDOUT
Q_{AIN}
Q_{AOUT}
OUT_DATA
OUT_VALID
OUT_ACCEPT
04
00
04
ff
00
00
04
ff
cc
00
04
ff
630
670
710
750
790
830
870
TIME

FIG. 5(B)

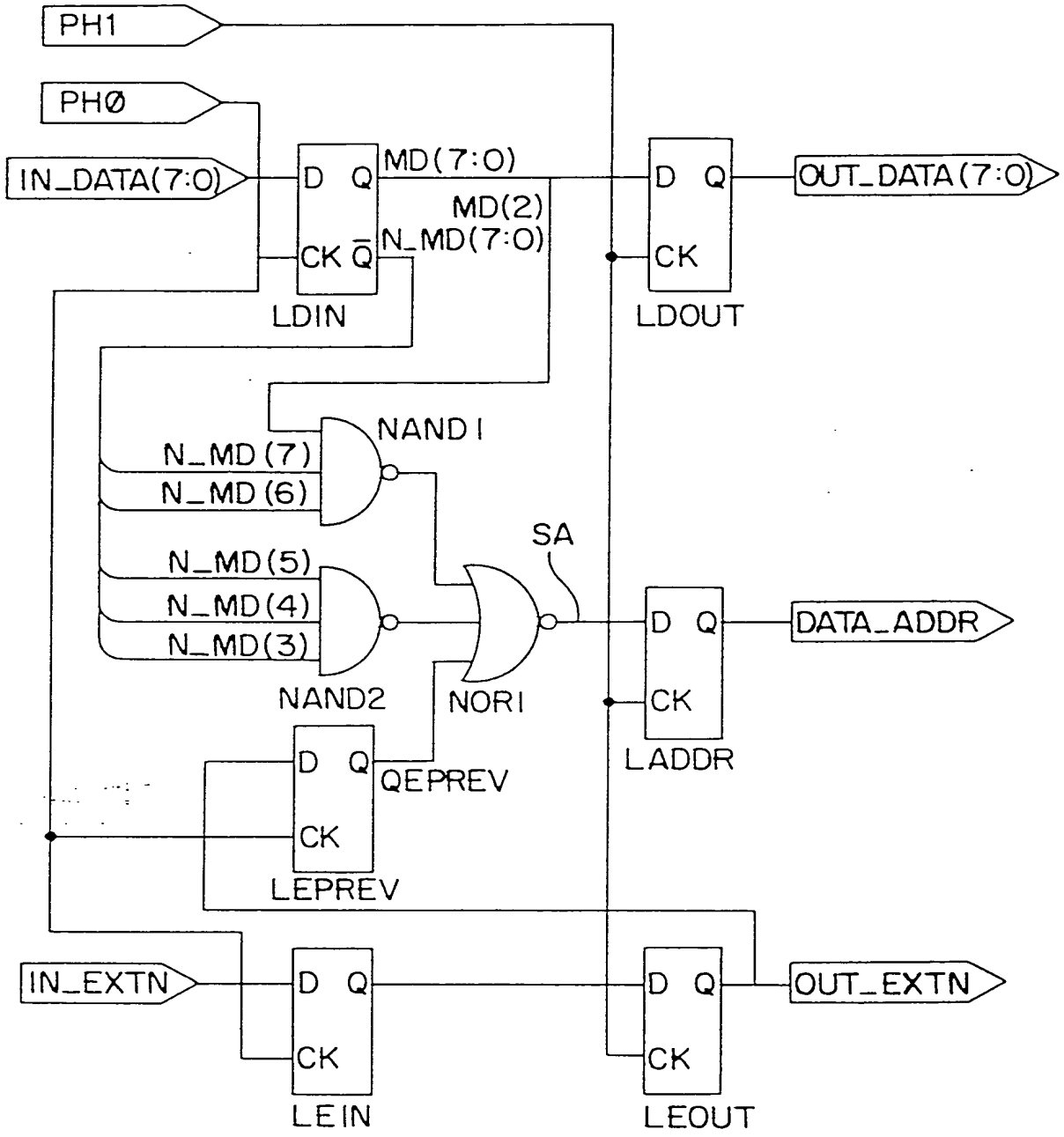

PH1
PHØ
IN_DATA(7:0)
D Q
CK Q̄
MD(7:0)
MD(2)
N_MD(7:0)
LDIN
D Q
CK
LDOUT
OUT_DATA(7:0)
NAND1
N_MD(7)
N_MD(6)
N_MD(5)
N_MD(4)
N_MD(3)
NAND2
NOR1
SA
D Q
CK
LADDR
DATA_ADDR
D Q
CK
QEPREV
LEPREV
IN_EXTN
D Q
CK
LEIN
D Q
CK
LEOUT
OUT_EXTN

FIG. 6

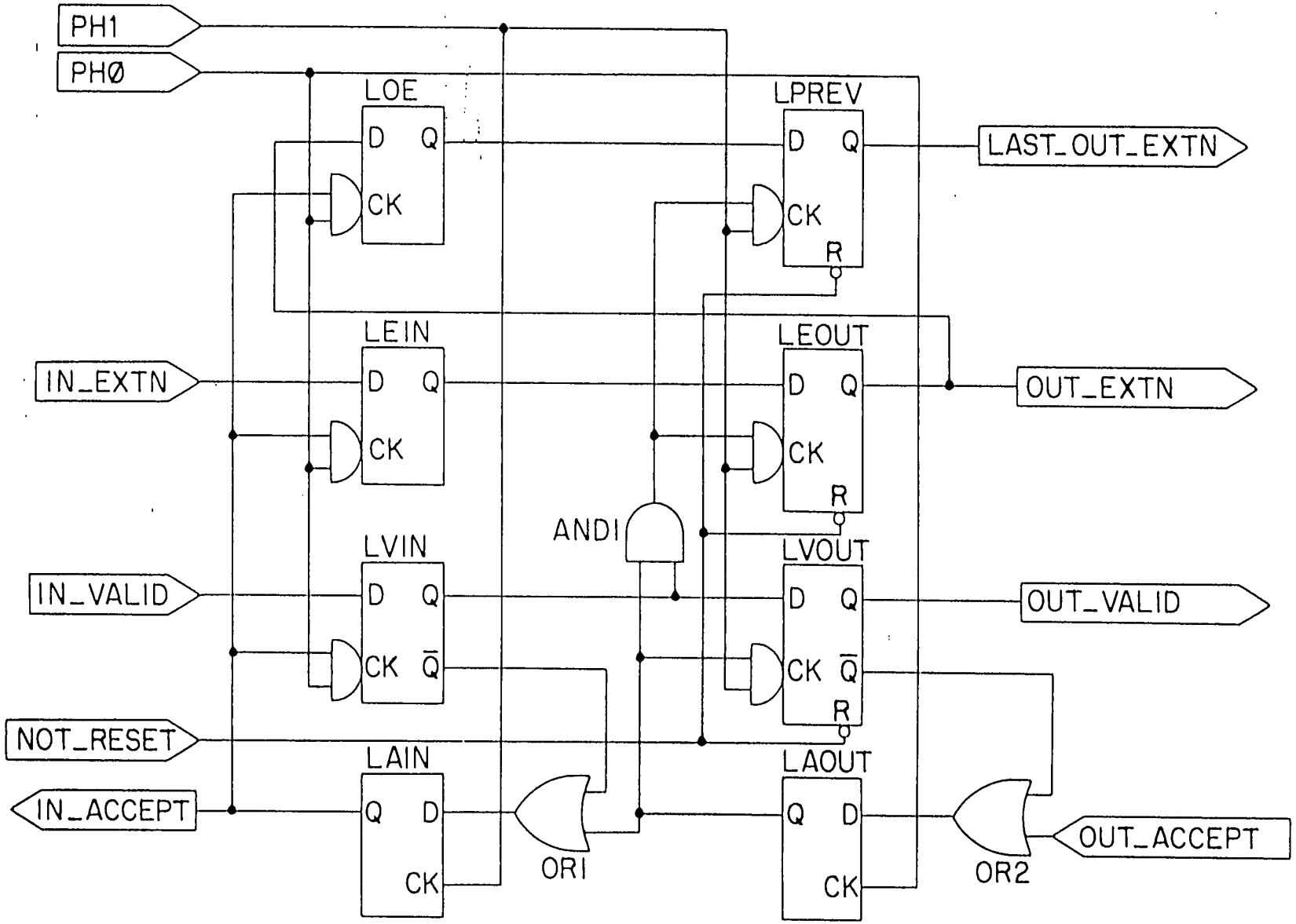

PH1
PHØ
LOE
D Q
CK
LPREV
D Q
CK
R
LAST_OUT_EXTN
LEIN
D Q
CK
LEOUT
D Q
CK
R
OUT_EXTN
IN_EXTN
ANDI
LVIN
D Q
CK Q̄
LVOUT
D Q
CK Q̄
R
OUT_VALID
IN_VALID
NOT_RESET
LAIN
Q D
CK
LAOUT
Q D
CK
IN_ACCEPT
OR1
OR2
OUT_ACCEPT

FIG. 7

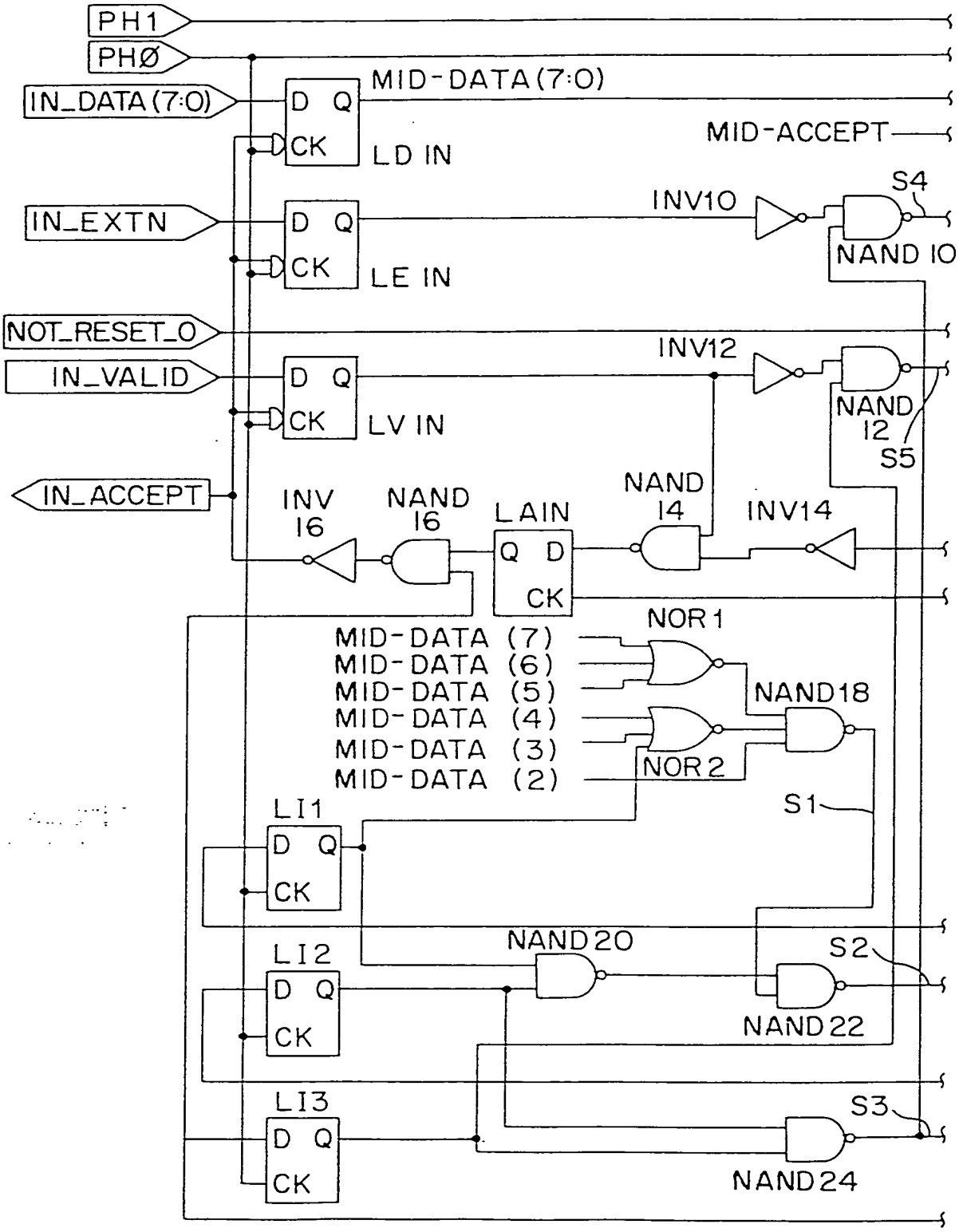

PH1
PHØ
IN_DATA(7:0)
MID-DATA(7:0)
MID-ACCEPT
D Q
CK
LD IN
IN_EXTN
LE IN
INV10
S4
NAND 10
NOT_RESET_O
IN_VALID
LV IN
INV12
NAND 12
S5
IN_ACCEPT
INV 16
NAND 16
LAIN
Q D
NAND 14
INV14
NOR 1
MID-DATA (7)
MID-DATA (6)
MID-DATA (5)
MID-DATA (4)
MID-DATA (3)
MID-DATA (2)
NAND18
NOR 2
S1
LI1
LI2
LI3
NAND20
S2
NAND 22
S3
NAND24

FIG. 8(A)

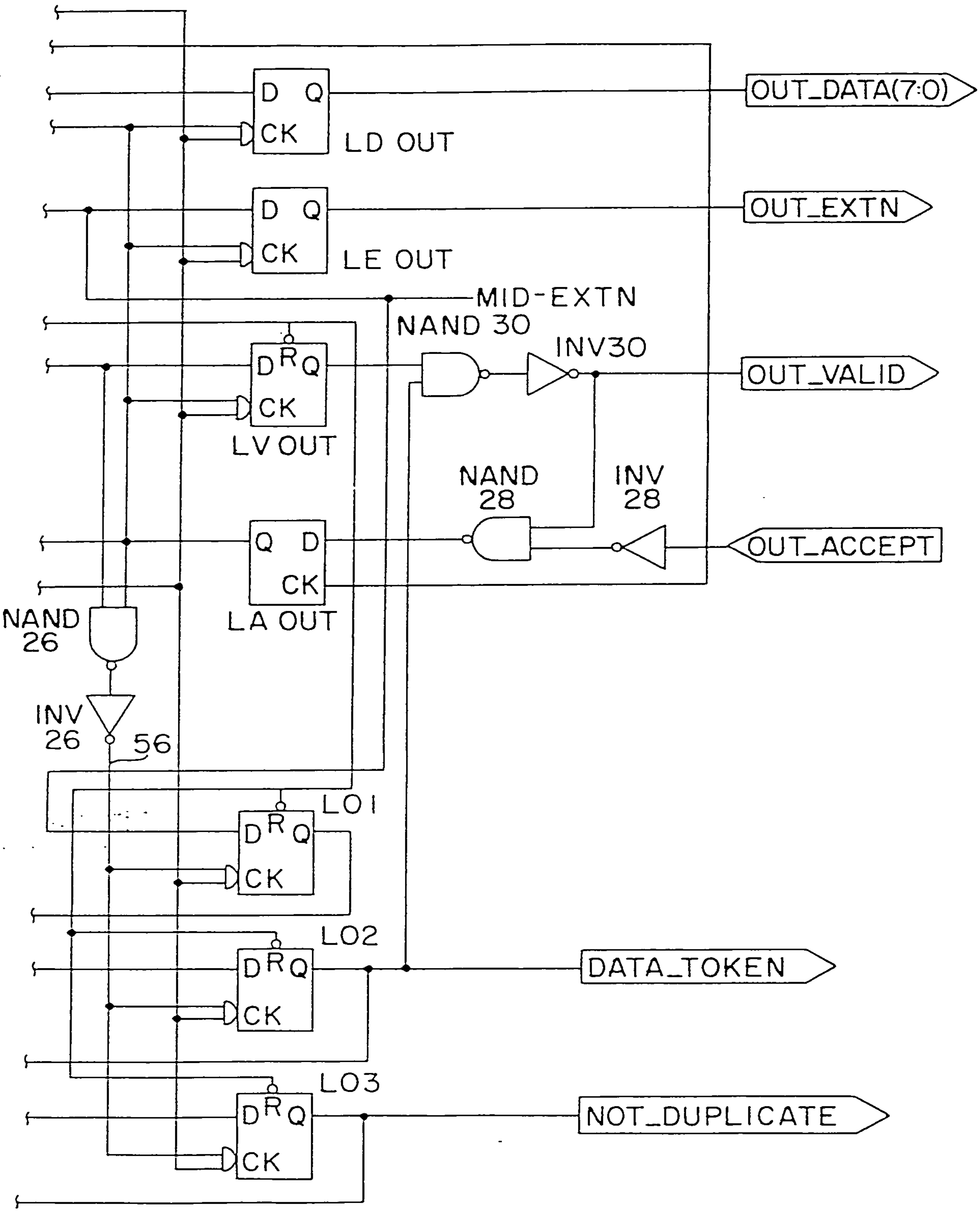

OUT_DATA(7:0)
D
Q
CK
LD OUT
OUT_EXTN
LE OUT
MID-EXTN
NAND 30
INV30
OUT_VALID
LV OUT
NAND
28
INV
28
OUT_ACCEPT
LA OUT
NAND
26
INV
26
56
LO1
LO2
DATA_TOKEN
LO3
NOT_DUPLICATE

FIG. 8(B)

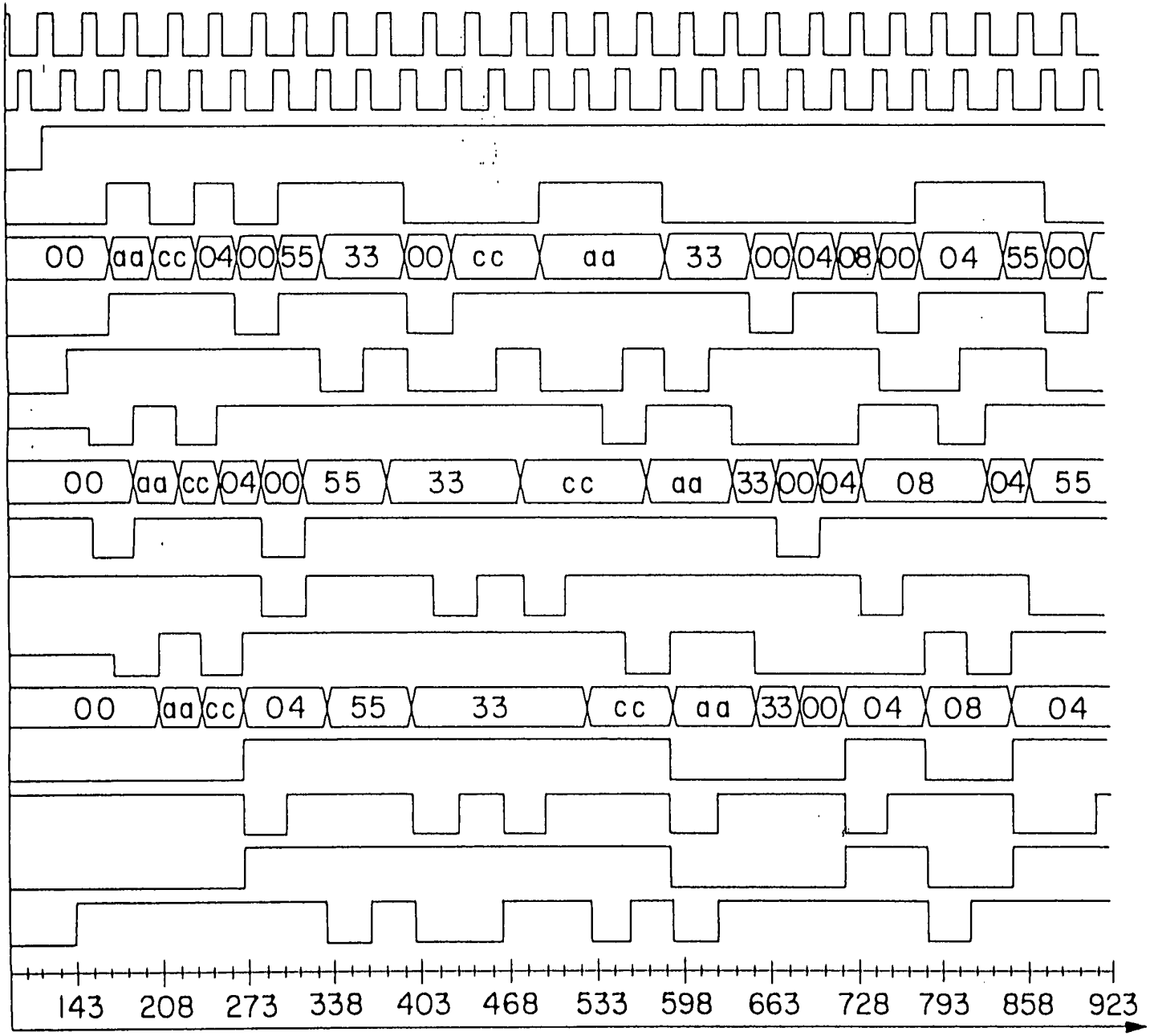

00 aa cc 04 00 55 33 00 cc aa 33 00 04 08 00 04 55 00
00 aa cc 04 00 55 33 cc aa 33 00 04 08 04 55
00 aa cc 04 55 33 cc aa 33 00 04 08 04
143 208 273 338 403 468 533 598 663 728 793 858 923
TIME

FIG. 9(A)

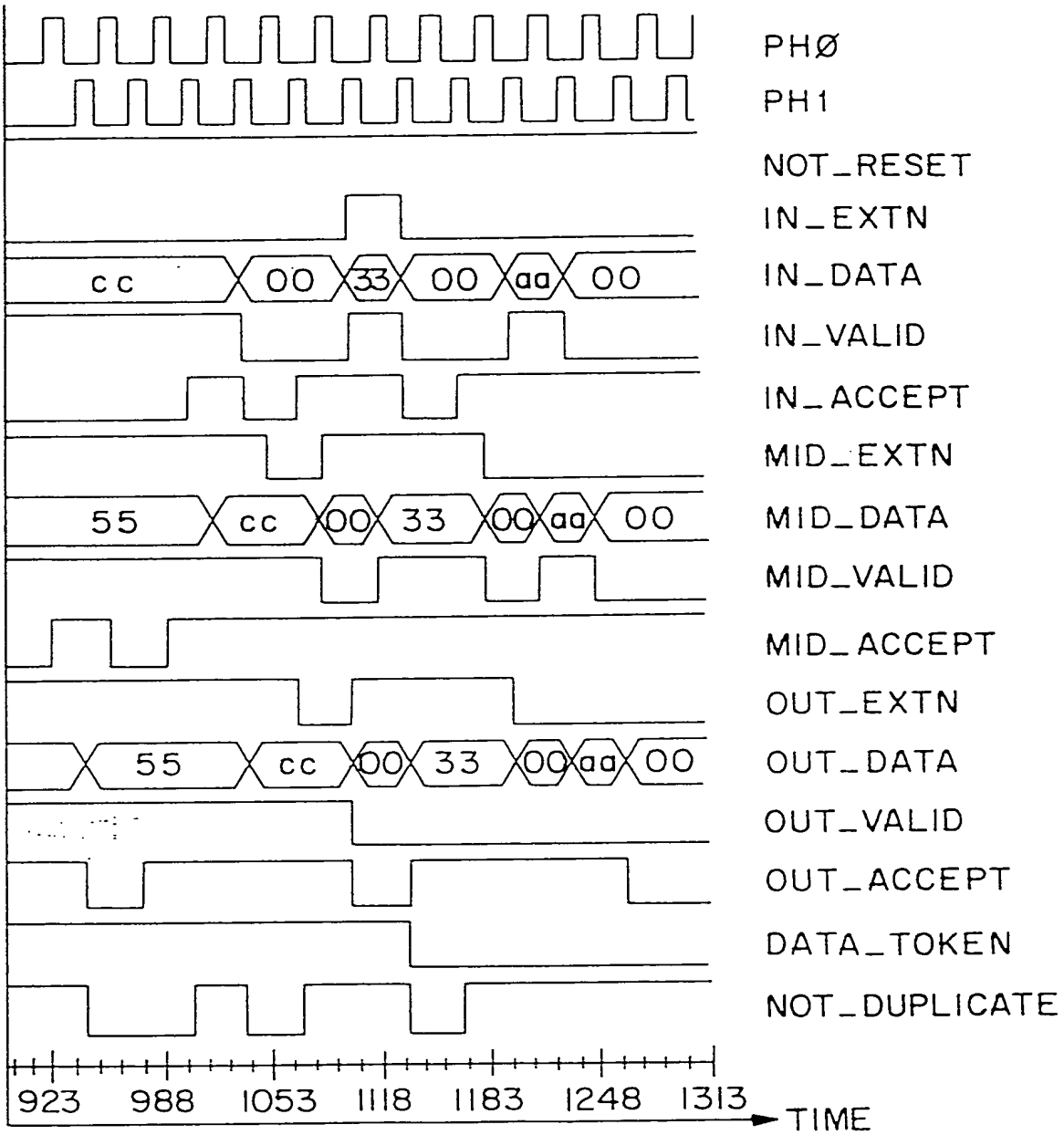


FIG. 9(B)

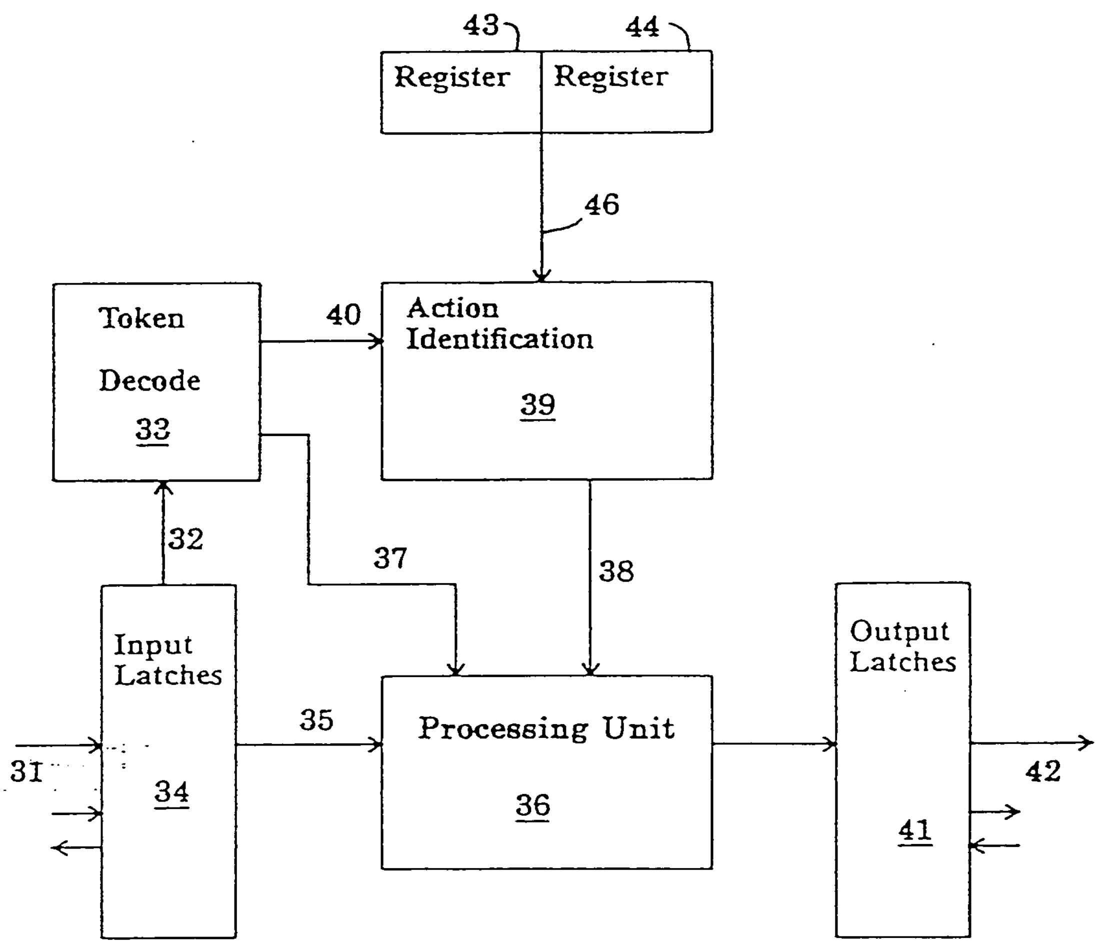


FIG. 10

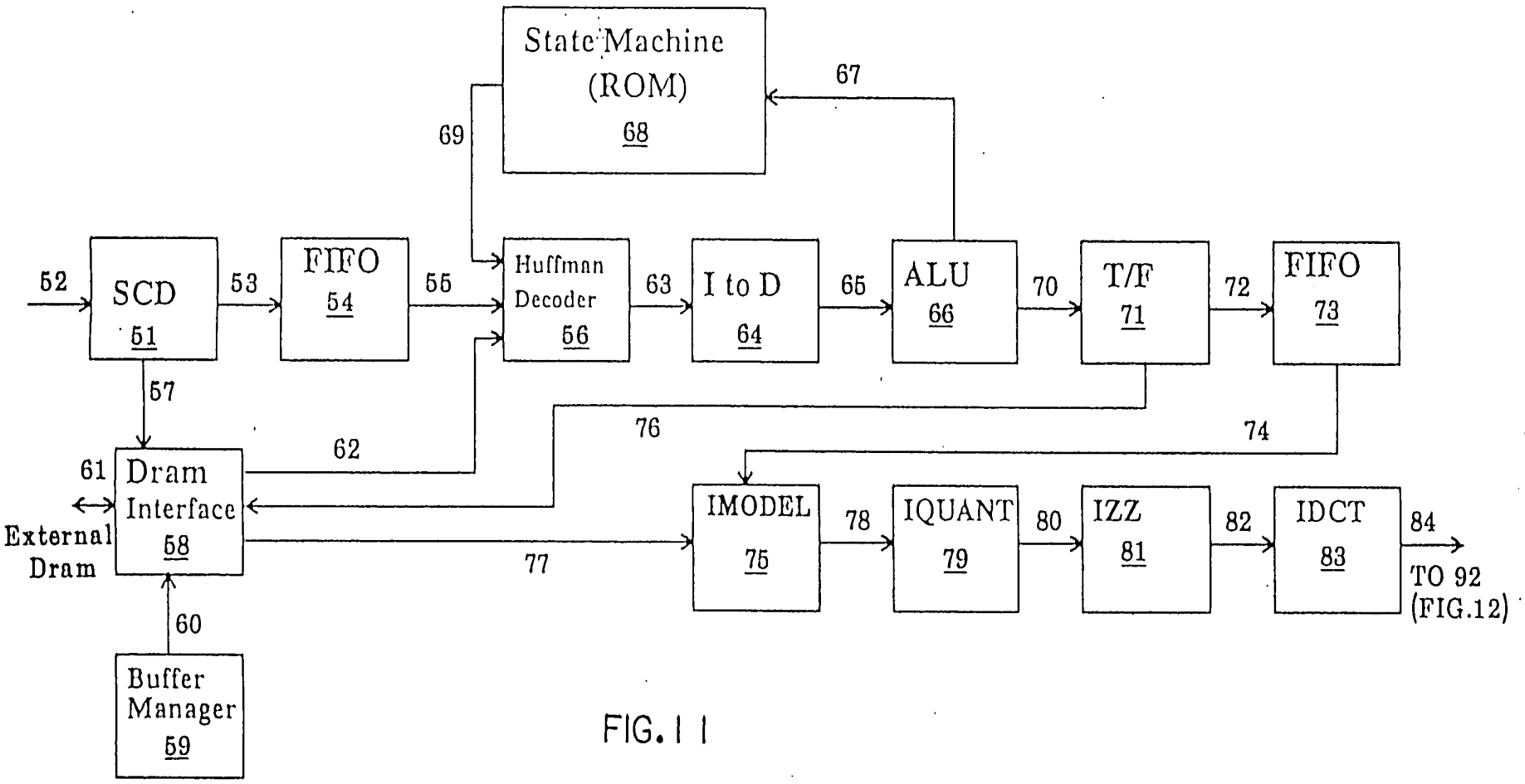

State Machine
(ROM)
68
67
69
52
SCD
51
53
FIFO
54
55
Huffman
Decoder
56
63
I to D
64
65
ALU
66
70
T/F
71
72
FIFO
73
57
76
74
61
Dram
Interface
58
62
External
Dram
IMODEL
75
78
IQUANT
79
80
IZZ
81
82
IDCT
83
84
TO 92
(FIG.12)
77
60
Buffer
Manager
59

FIG. 11

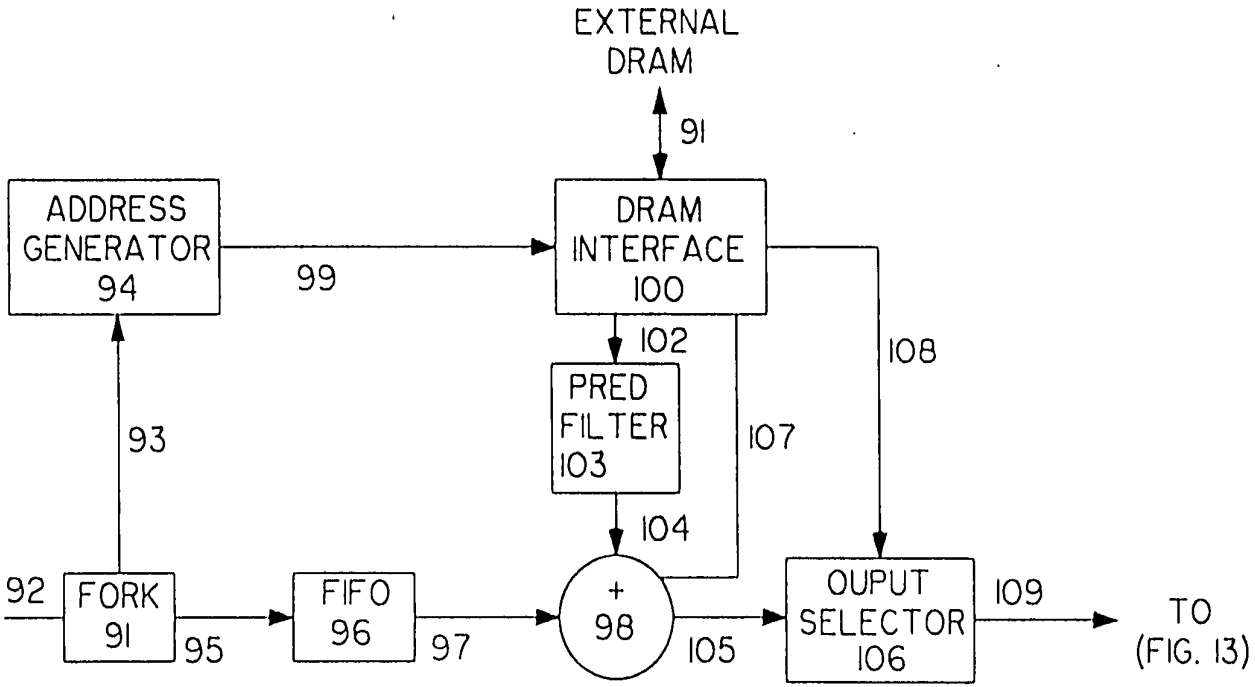


FIG. 12

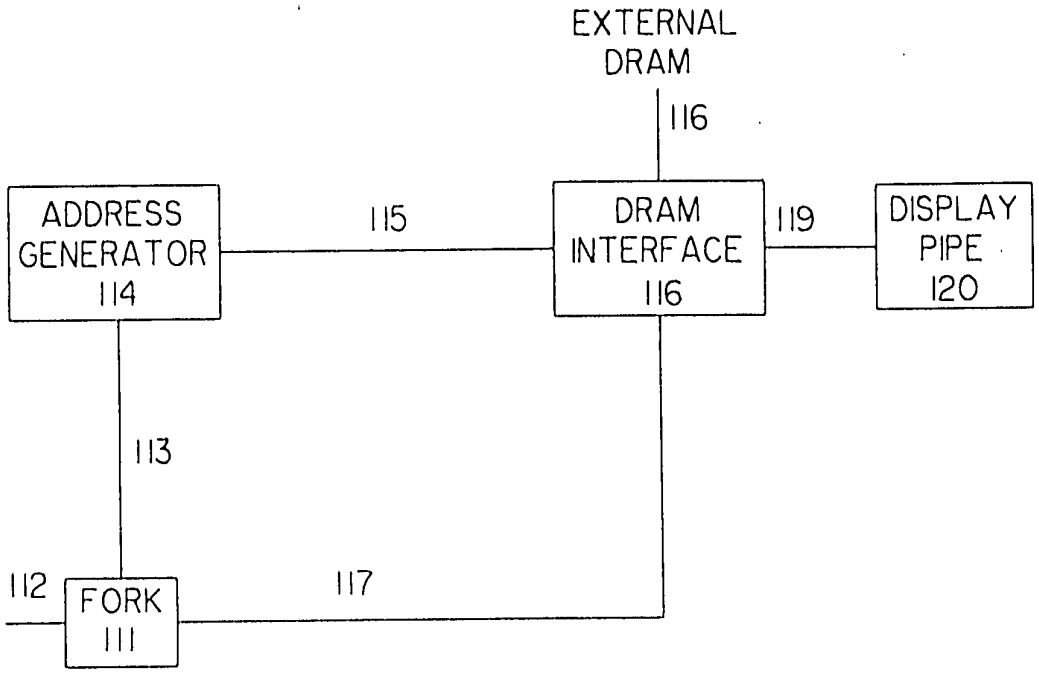

EXTERNAL
DRAM
116
ADDRESS
GENERATOR
114
115
DRAM
INTERFACE
116
119
DISPLAY
PIPE
120
113
112
FORK
111
117

FIG. 13

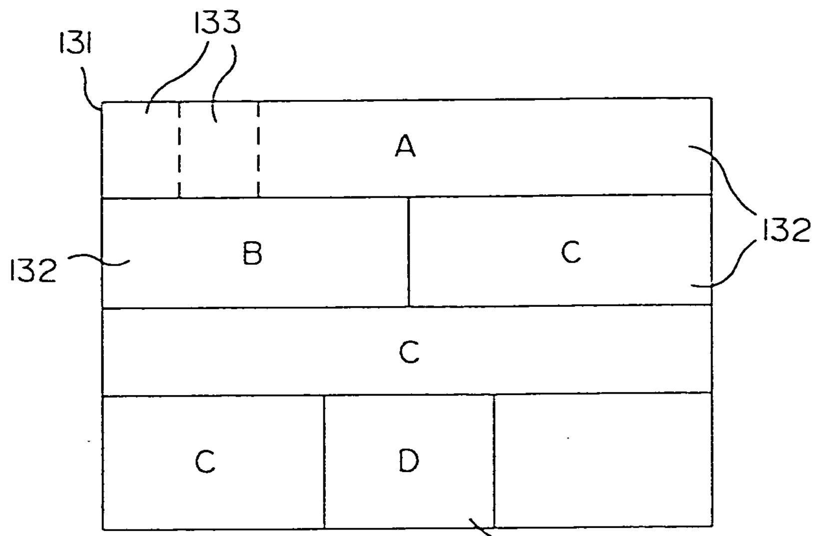


FIG. 14a

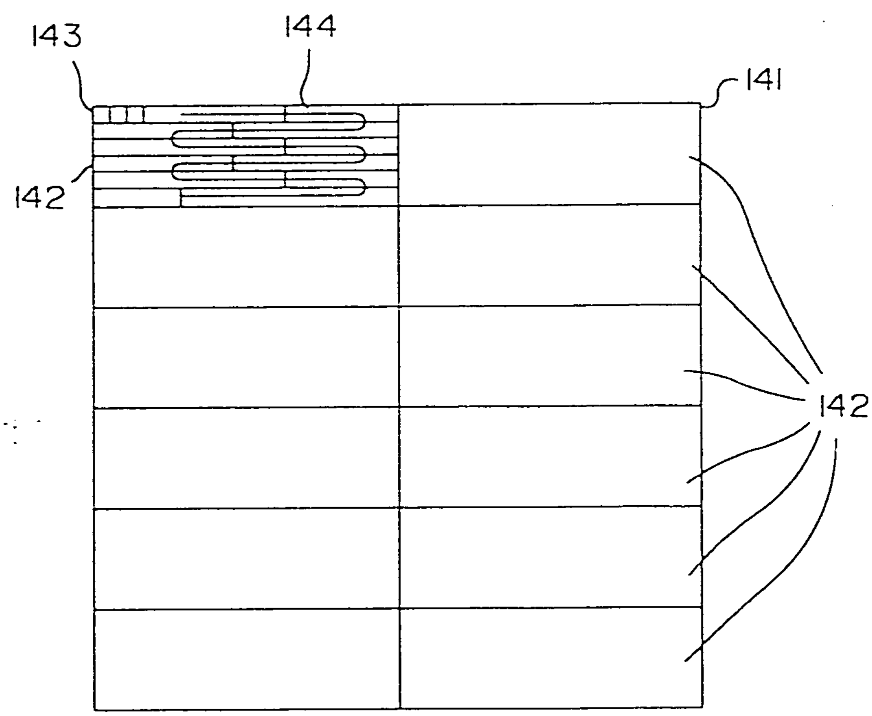


FIG. 14b

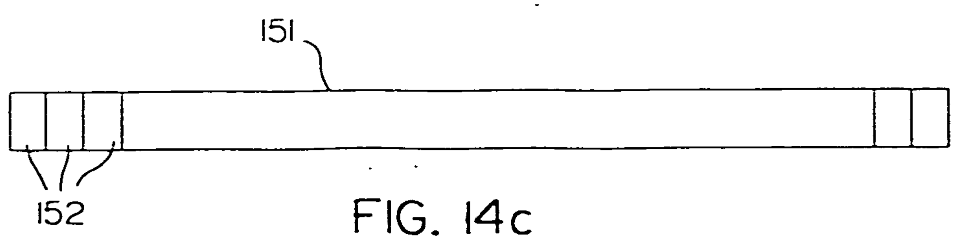


FIG. 14c

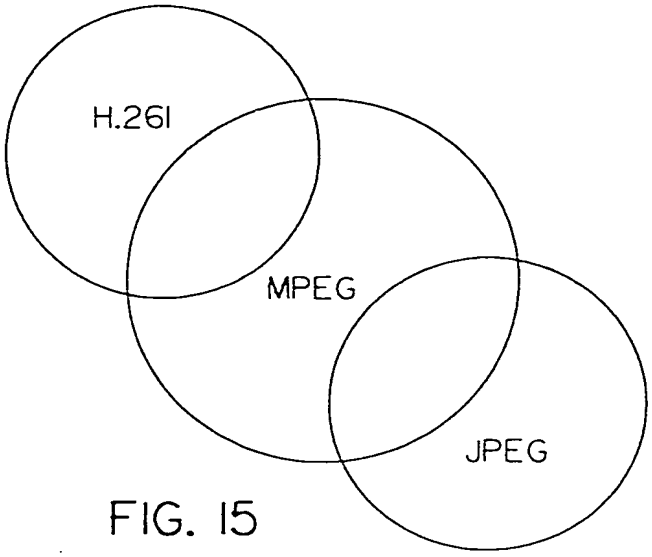

H.261
MPEG
JPEG

FIG. 15

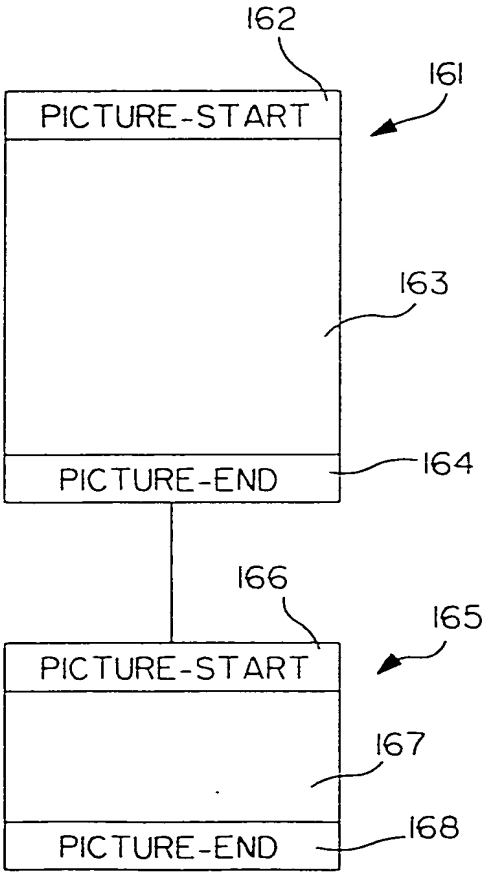

162
161
PICTURE-START
163
PICTURE-END
164
166
165
PICTURE-START
167
PICTURE-END
168

FIG. 16

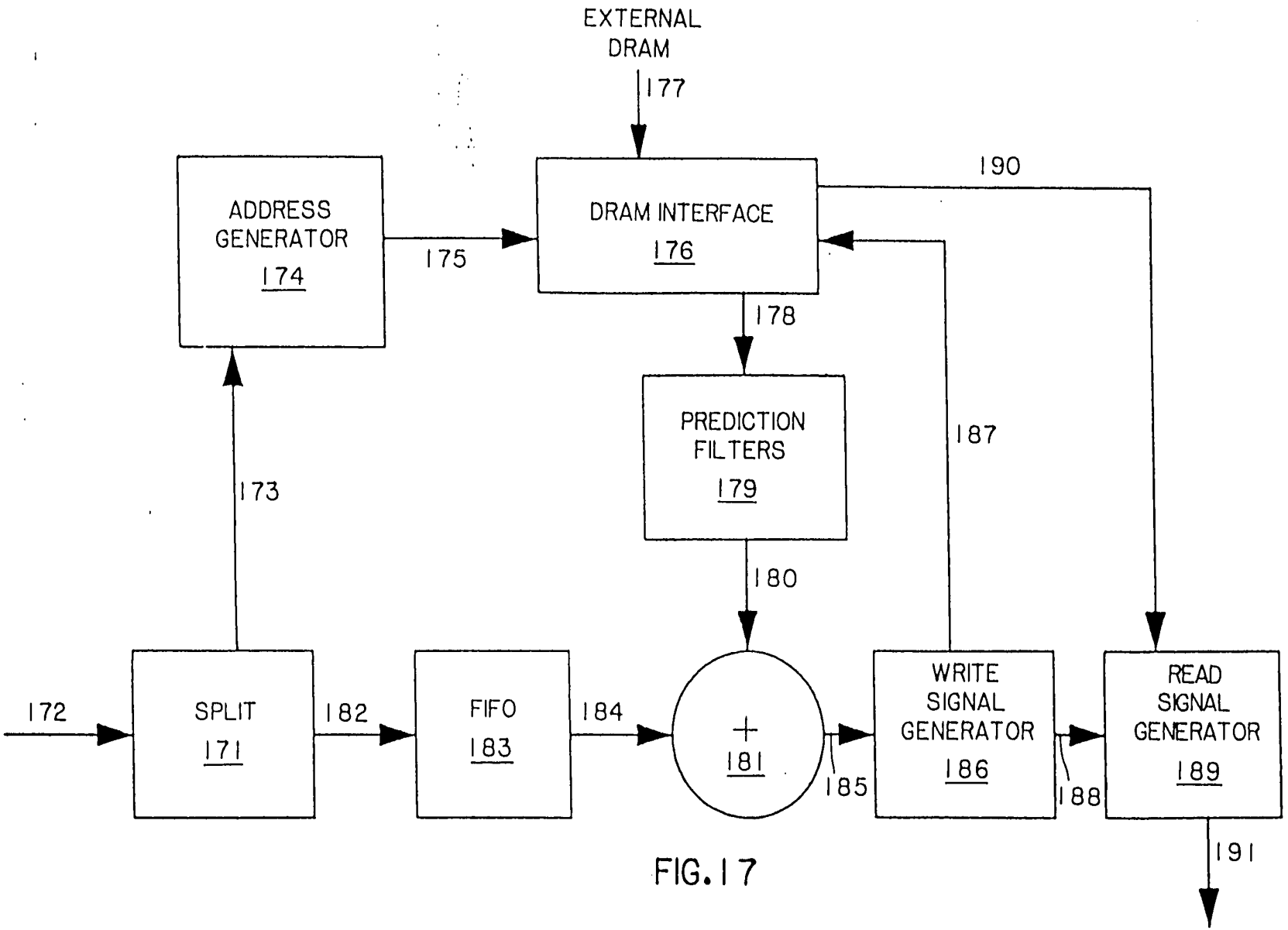

EXTERNAL
DRAM
177
ADDRESS
GENERATOR
174
175
DRAM INTERFACE
176
190
178
PREDICTION
FILTERS
179
187
173
180
172
SPLIT
171
182
FIFO
183
184
+
181
185
WRITE
SIGNAL
GENERATOR
186
188
READ
SIGNAL
GENERATOR
189
191

FIG. 17

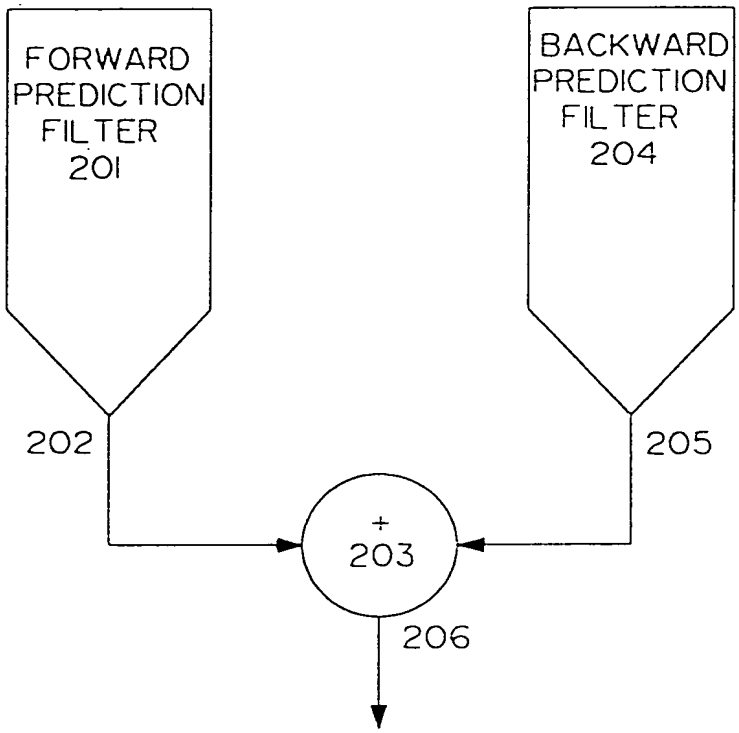


FIG. 18

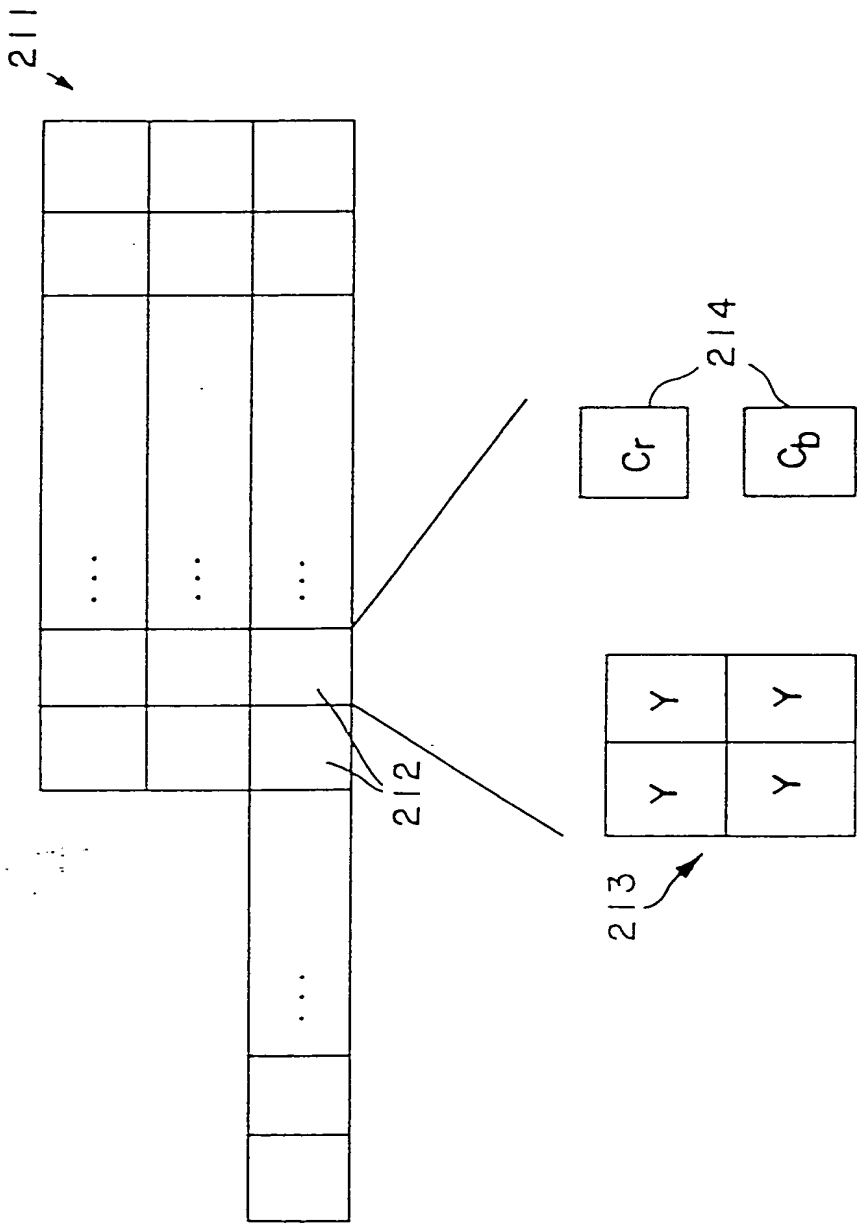


FIG. 19

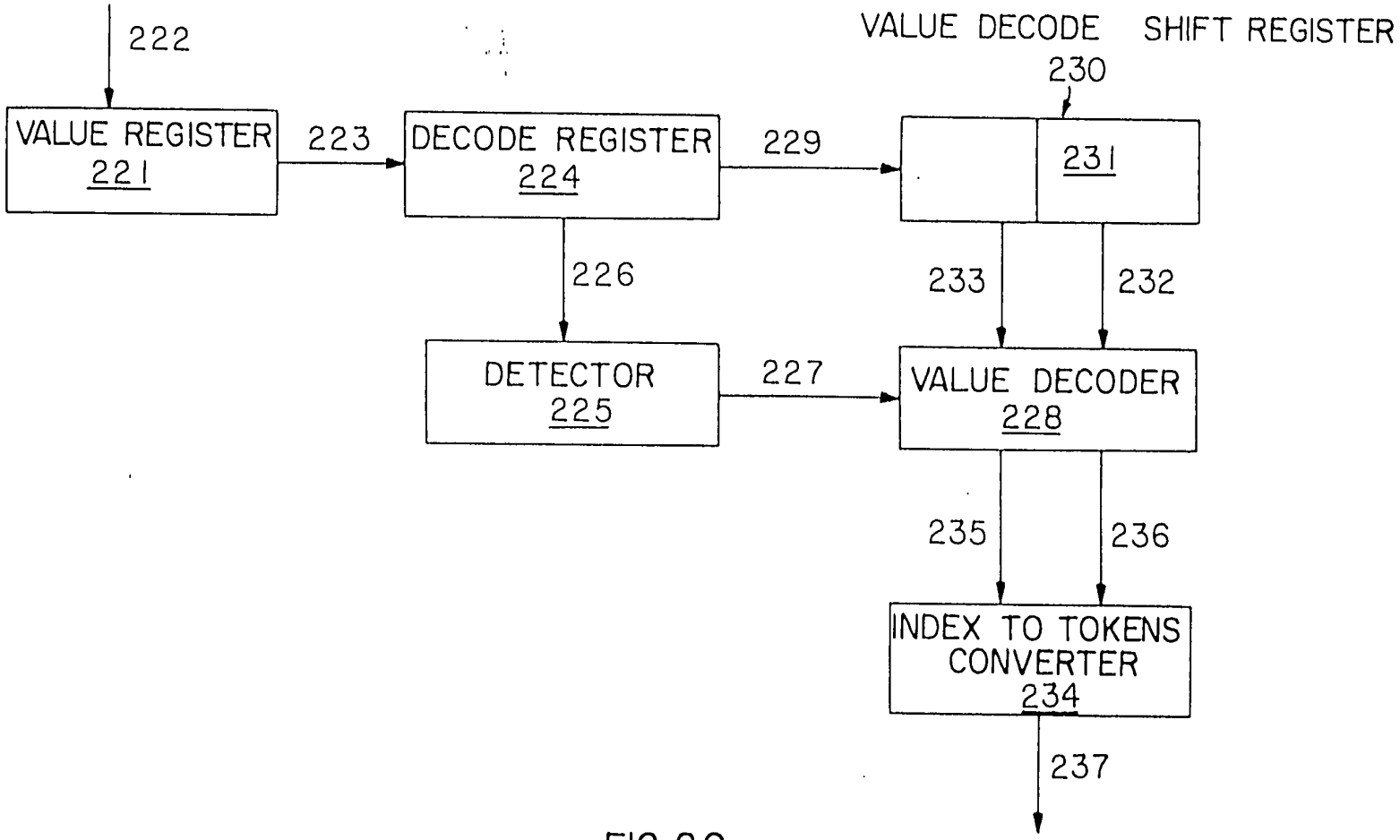

222
VALUE REGISTER
221
223
DECODE REGISTER
224
229
VALUE DECODE SHIFT REGISTER
230
231
226
233
232
DETECTOR
225
227
VALUE DECODER
228
235
236
INDEX TO TOKENS
CONVERTER
234
237

FIG.20

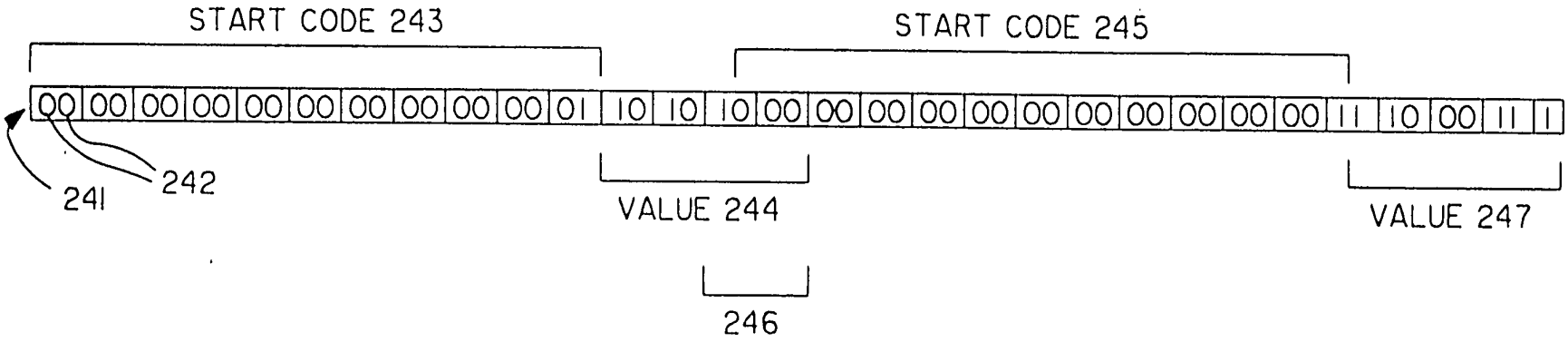

START CODE 243
START CODE 245
00 00 00 00 00 00 00 00 00 00 01 10 10 10 00 00 00 00 00 00 00 00 00 00 00 11 10 00 11 1
241
242
VALUE 244
VALUE 247
246

FIG. 21

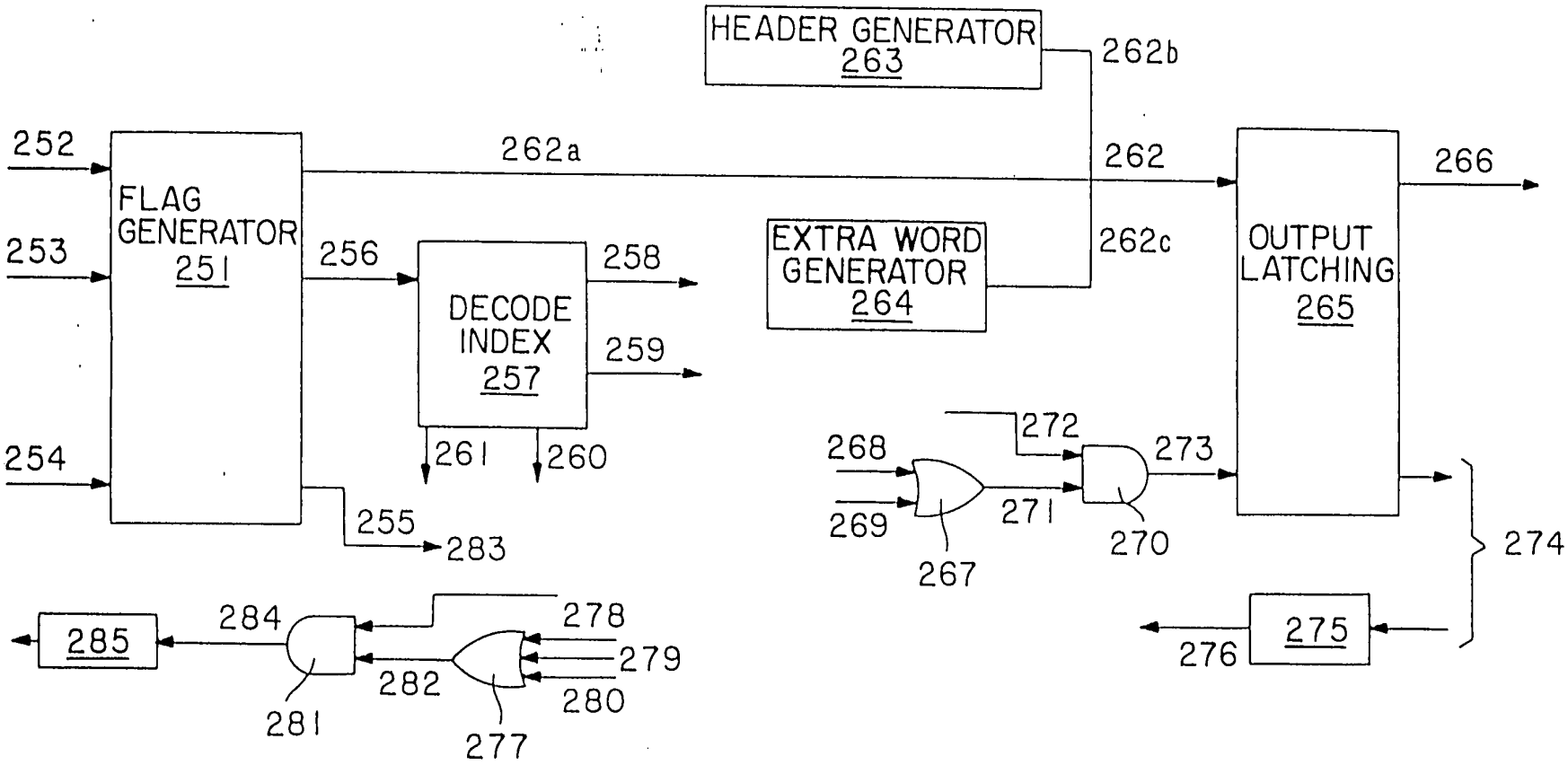

HEADER GENERATOR
263
262b
252
FLAG
GENERATOR
251
262a
262
266
253
256
DECODE
INDEX
257
258
259
EXTRA WORD
GENERATOR
264
262c
OUTPUT
LATCHING
265
254
261
260
255
283
272
268
273
269
271
270
267
274
285
284
278
279
282
280
281
277
275
276

FIG.22

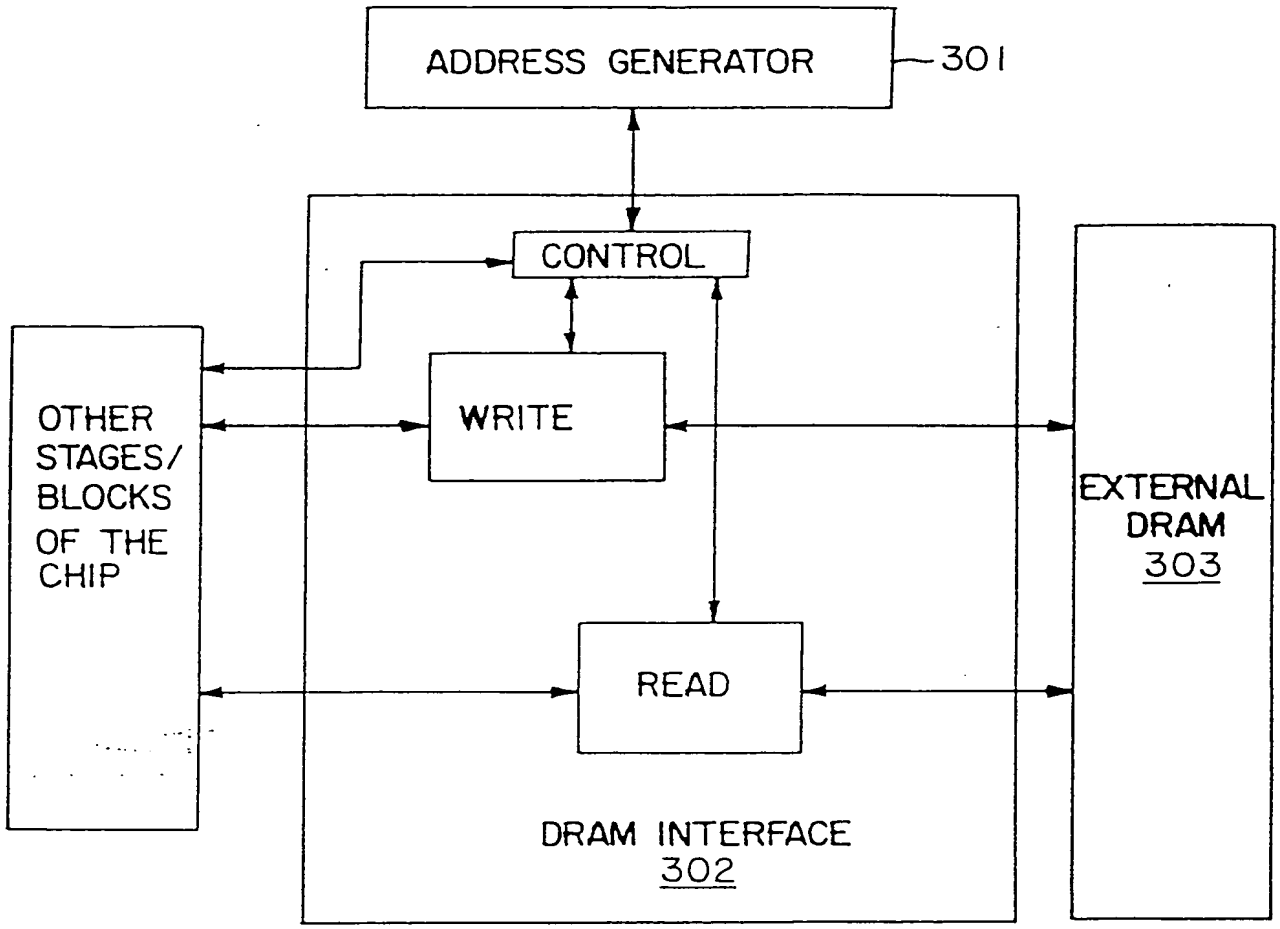

ADDRESS GENERATOR
301
CONTROL
OTHER STAGES/ BLOCKS OF THE CHIP
WRITE
READ
EXTERNAL DRAM 303
DRAM INTERFACE 302

FIG.23

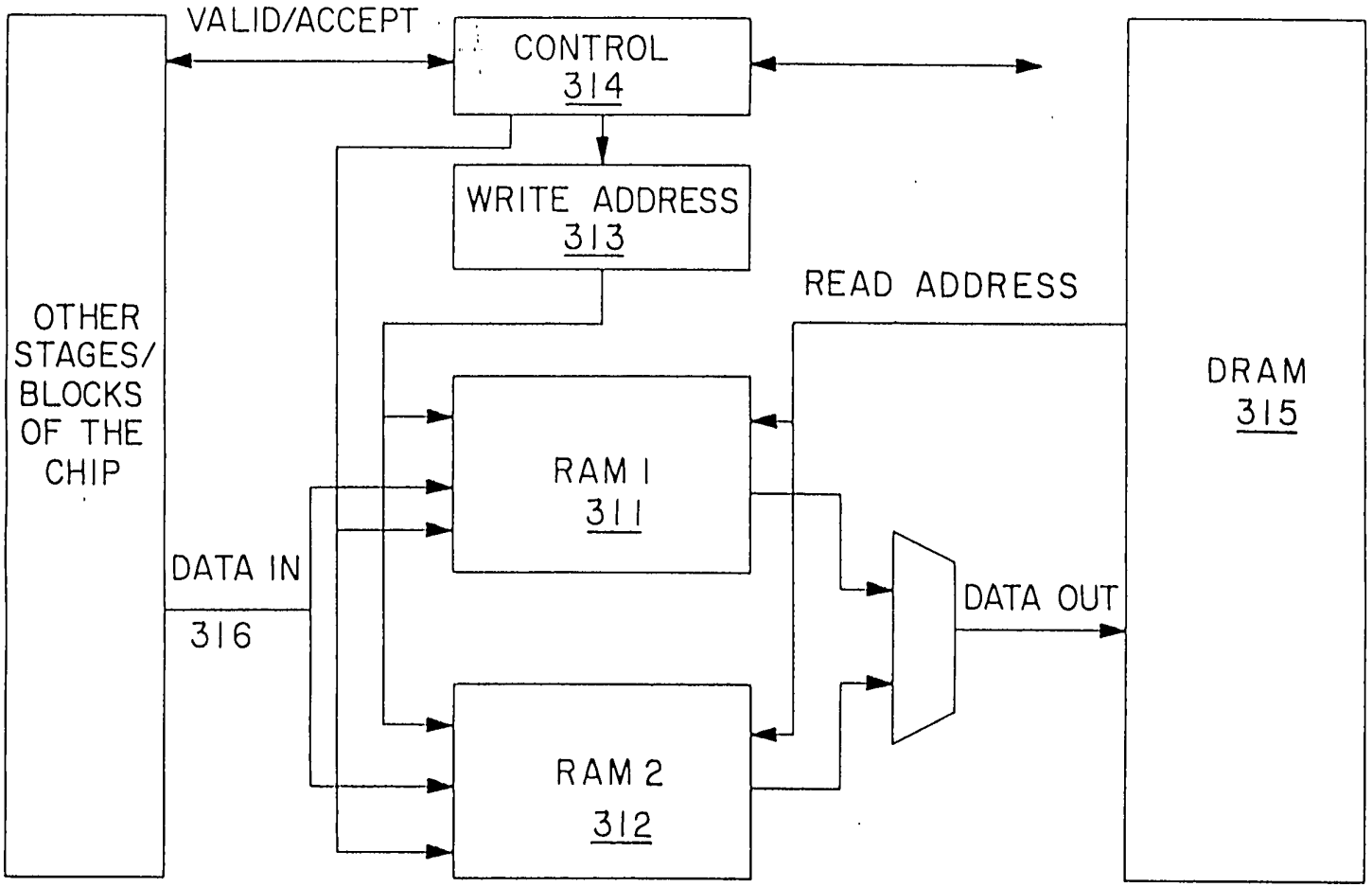

VALID/ACCEPT
CONTROL
314
WRITE ADDRESS
313
READ ADDRESS
OTHER
STAGES/
BLOCKS
OF THE
CHIP
DRAM
315
RAM 1
311
DATA IN
316
DATA OUT
RAM 2
312

FIG.24

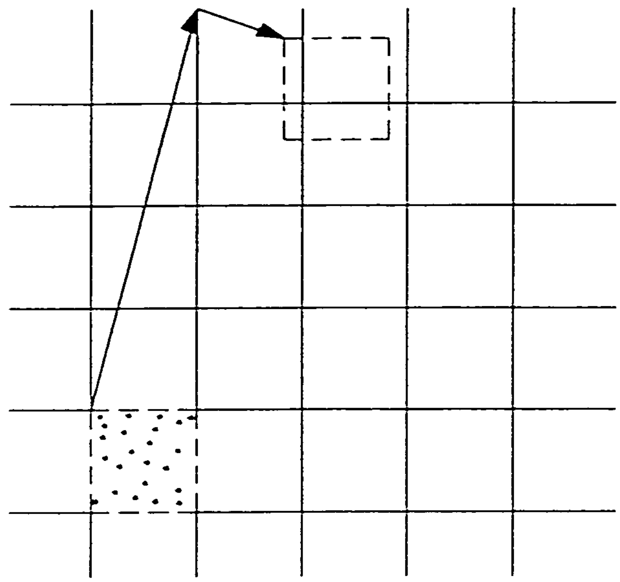


FIG. 25

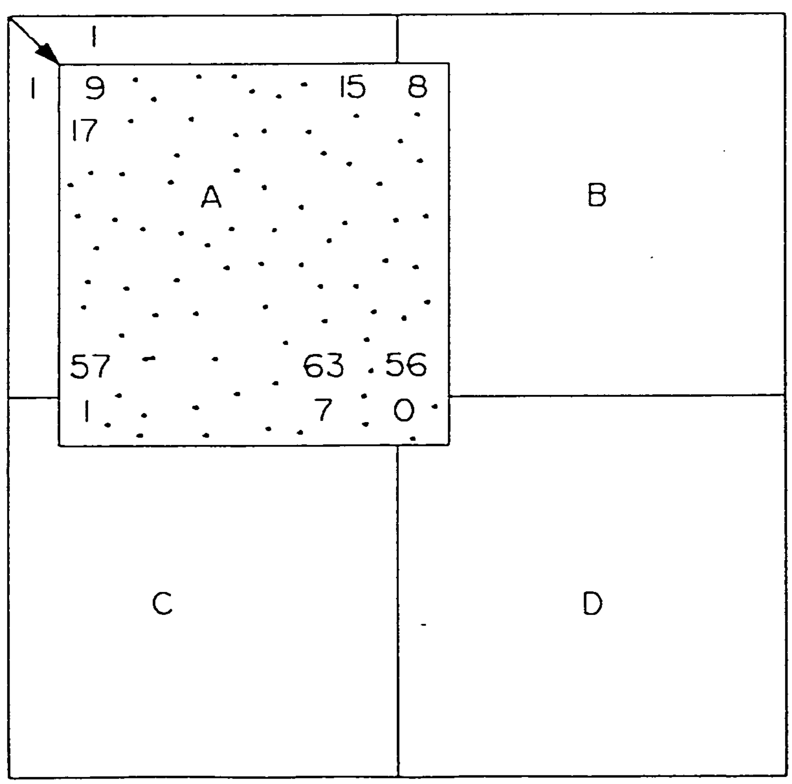


FIG. 26

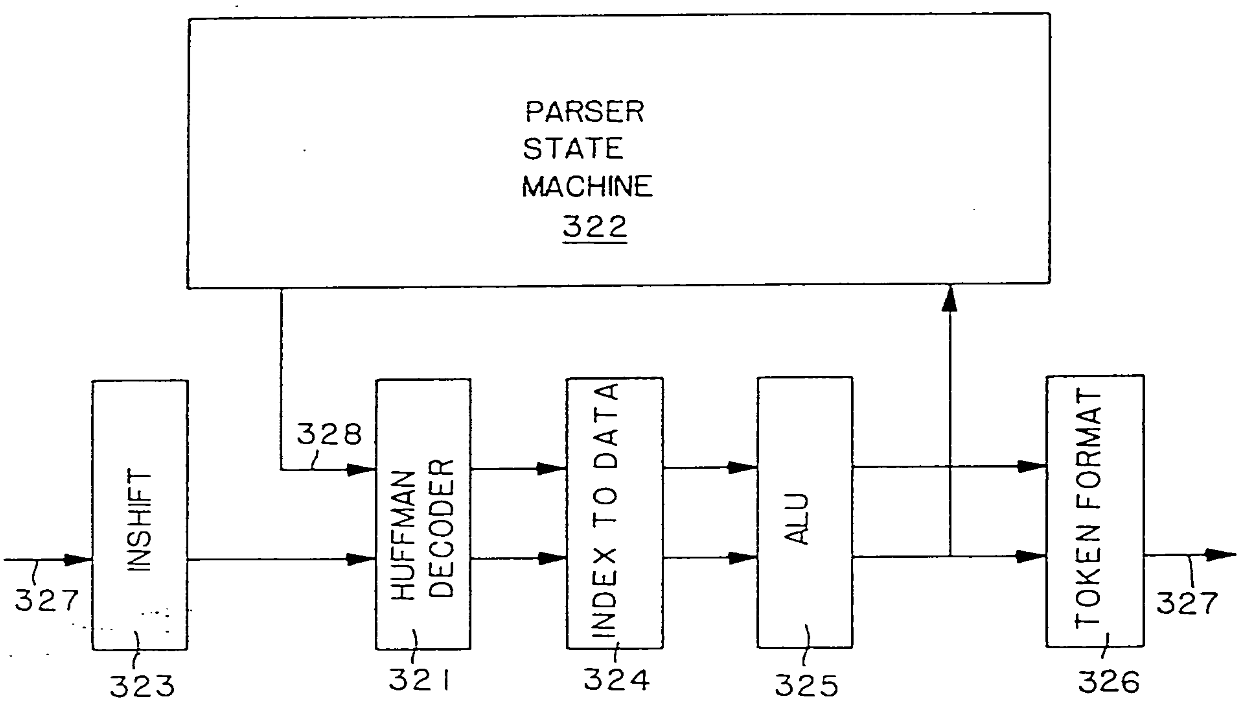

PARSER
STATE
MACHINE
322
INSHIFT
HUFFMAN
DECODER
INDEX TO DATA
ALU
TOKEN FORMAT
328
327
323
321
324
325
326
327

FIG.27

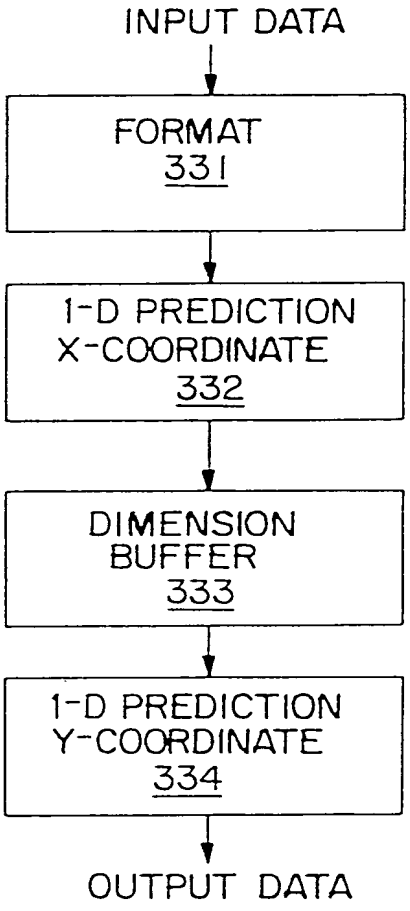

INPUT DATA
FORMAT
331
1-D PREDICTION
X-COORDINATE
332
DIMENSION
BUFFER
333
1-D PREDICTION
Y-COORDINATE
334
OUTPUT DATA

FIG.28

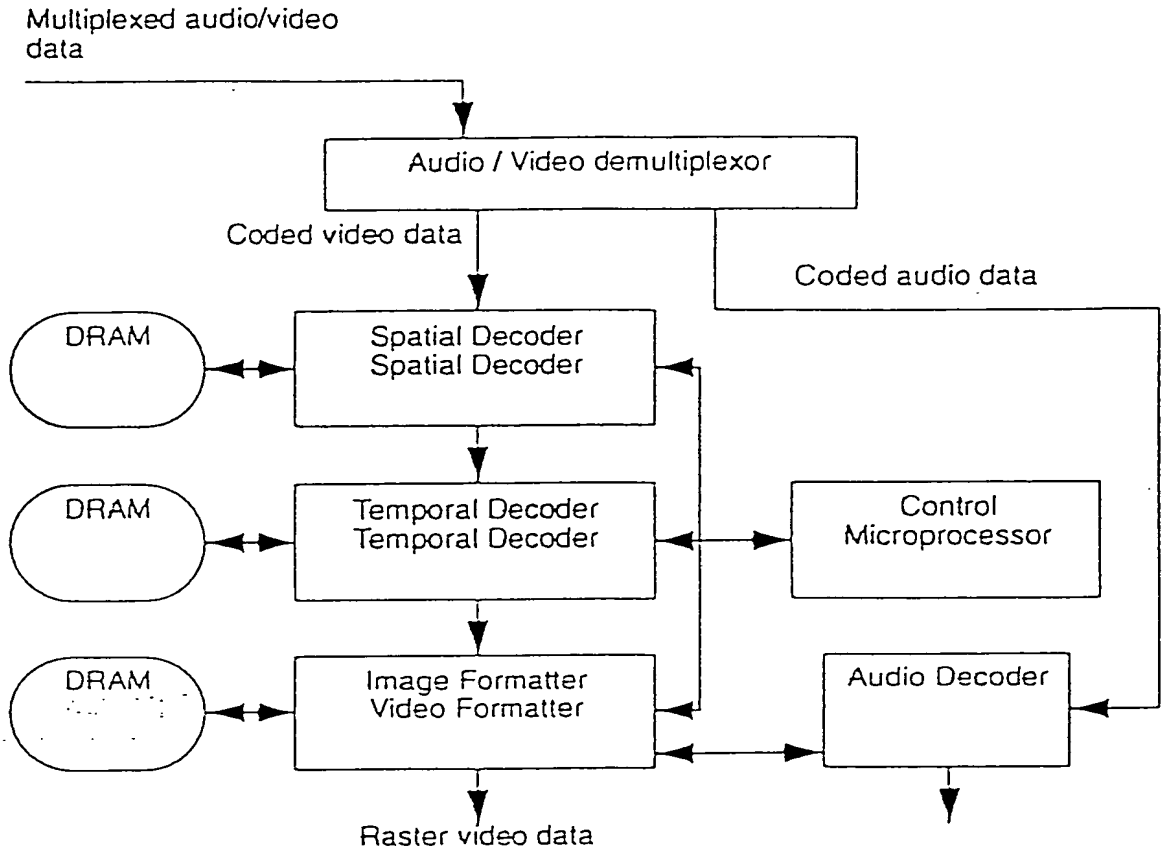


FIG.29

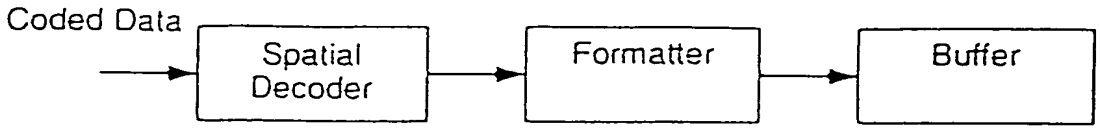


FIG.30

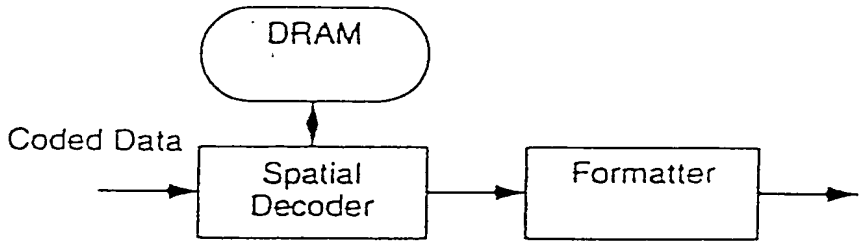


FIG.31

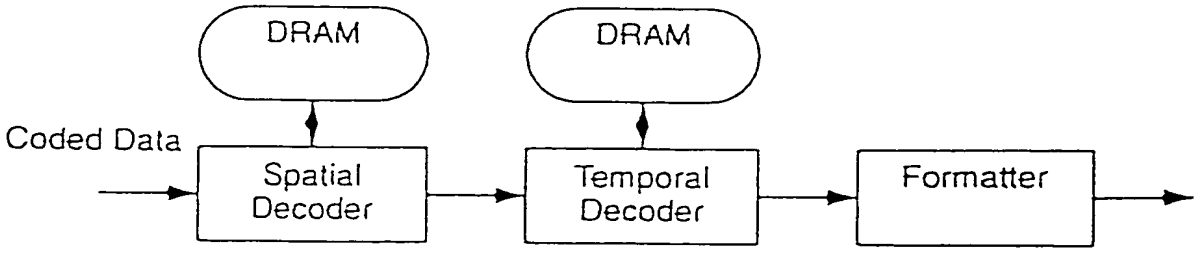


FIG.32

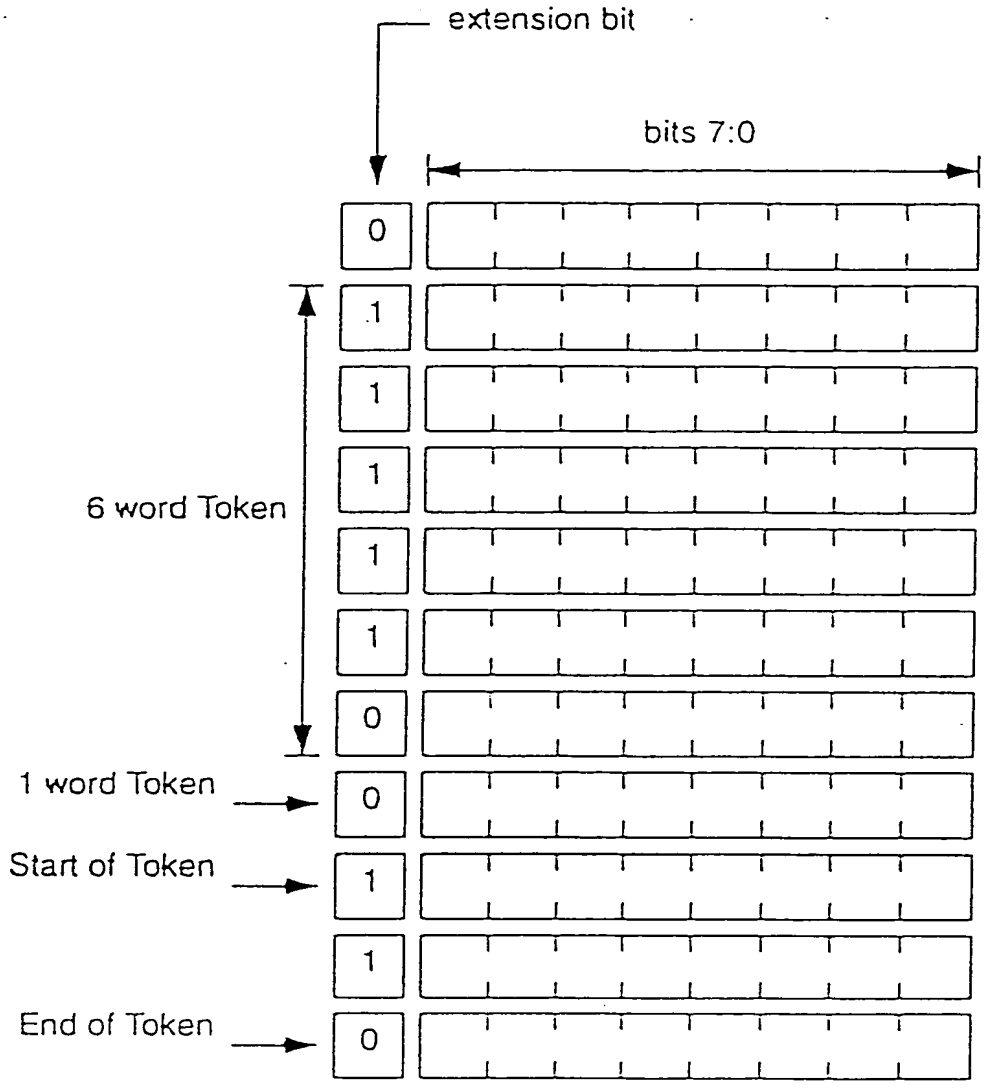

extension bit
bits 7:0
0
1
1
1
1
1
0
6 word Token
1 word Token
0
Start of Token
1
1
End of Token
0

FIG.33

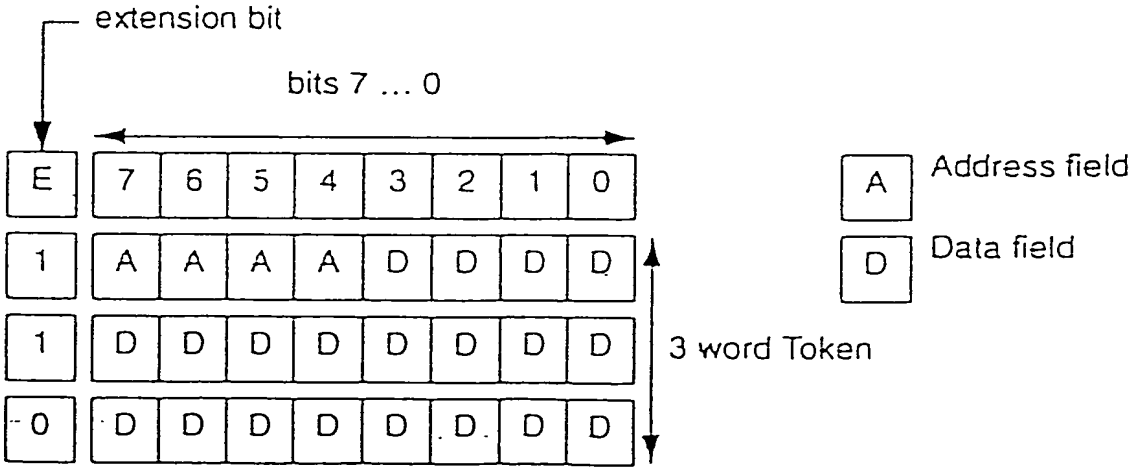

extension bit
bits 7 ... 0
E
7
6
5
4
3
2
1
0
1
A
A
A
A
D
D
D
D
1
D
D
D
D
D
D
D
D
0
D
D
D
D
D
D
D
D
3 word Token
A
Address field
D
Data field

FIG.34

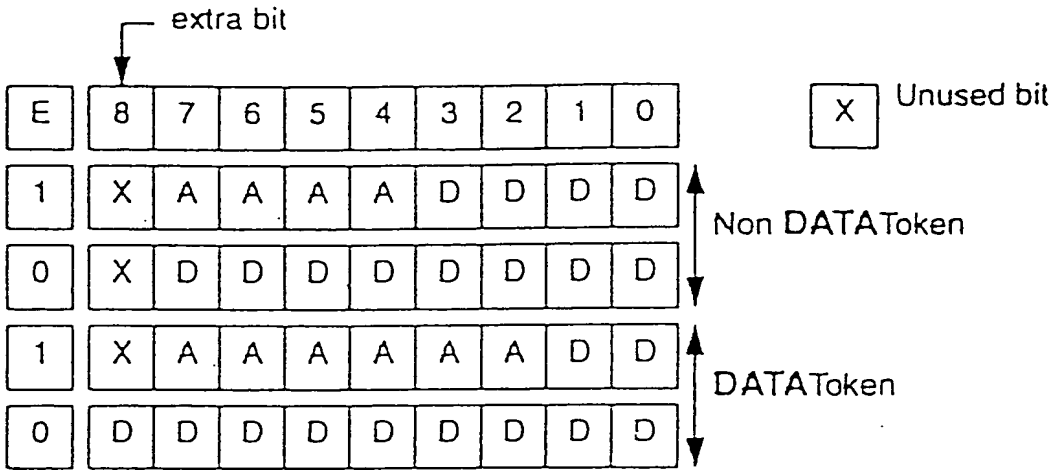
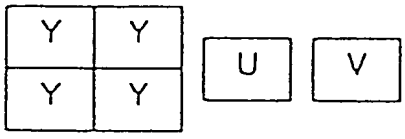


FIG.35



MPEG 4:2:0
macroblock

FIG.36A

JPEG 2:1:1
macroblock

FIG.36B

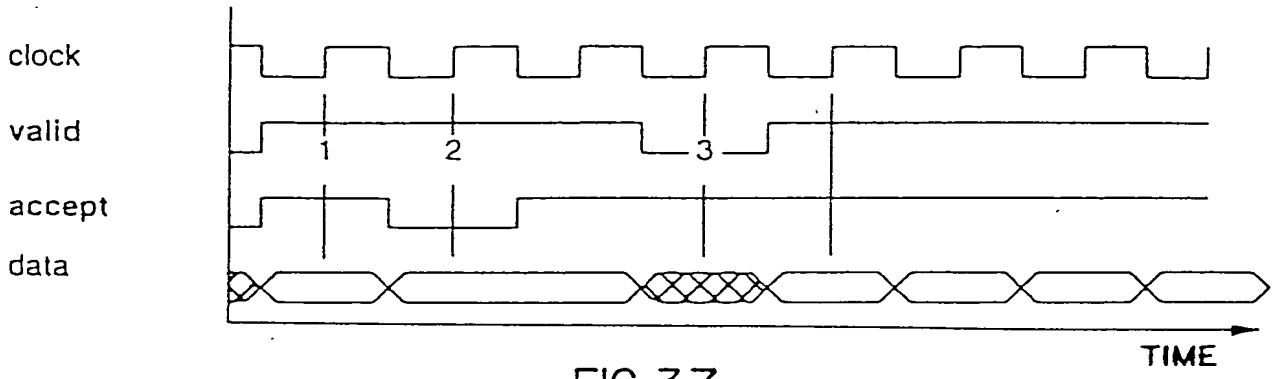

clock
valid
accept
data
1
2
3
TIME

FIG.37

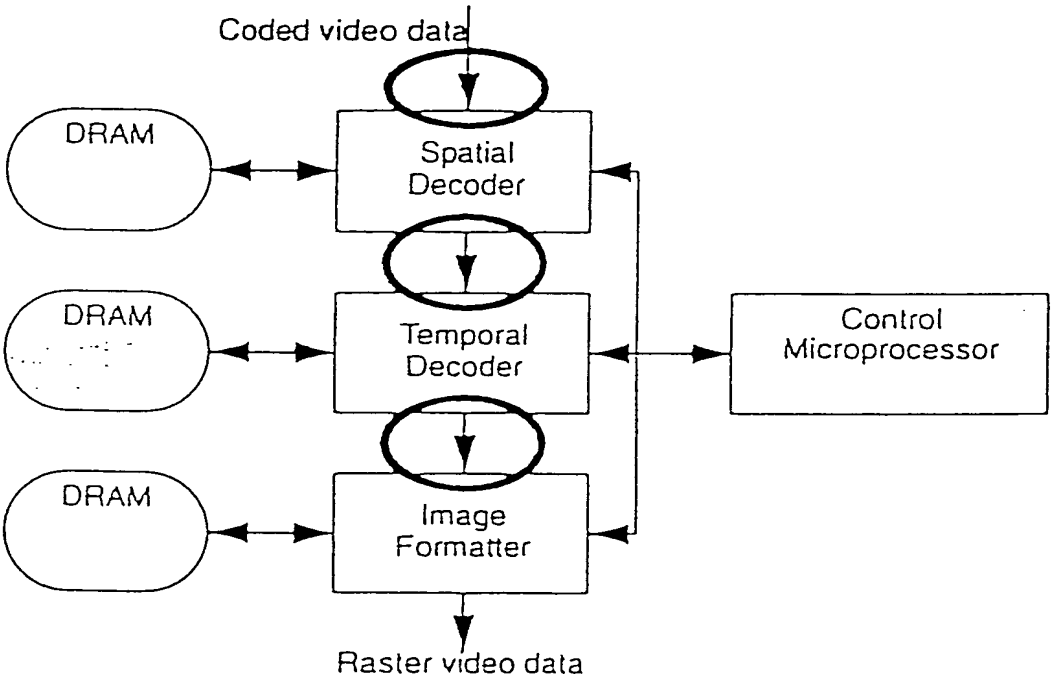

Coded video data
DRAM
Spatial Decoder
DRAM
Temporal Decoder
Control Microprocessor
DRAM
Image Formatter
Raster video data

FIG.38

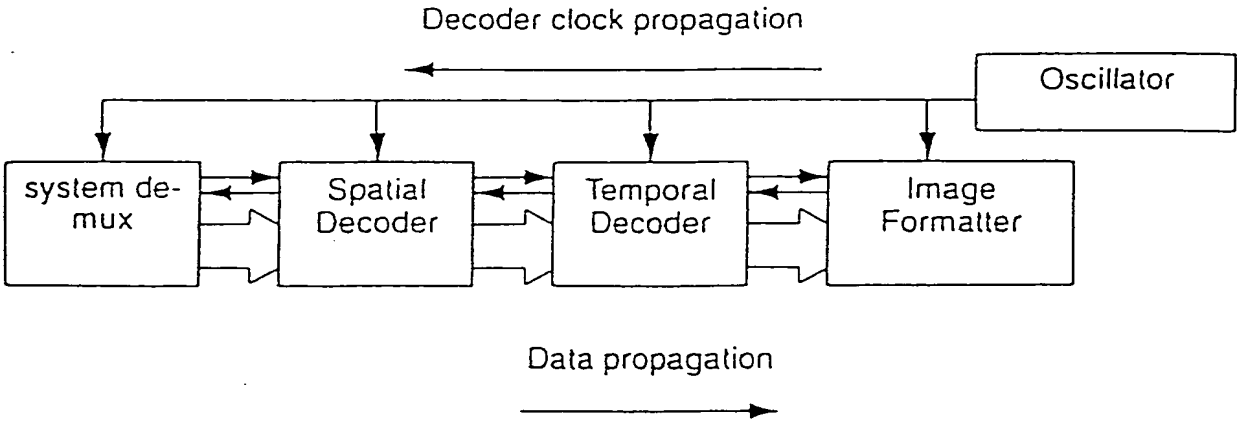

Decoder clock propagation
Oscillator
system de-mux
Spatial Decoder
Temporal Decoder
Image Formatter
Data propagation

FIG.39

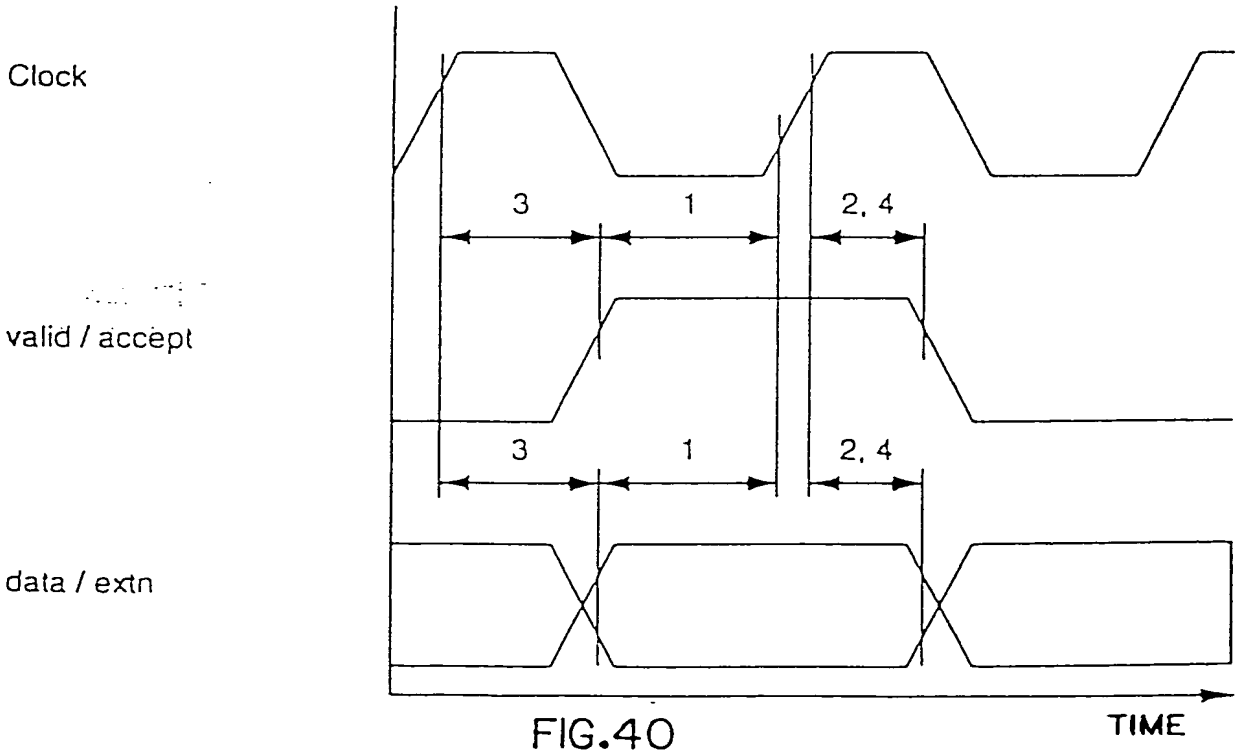

Clock
valid / accept
data / extn
3
1
2, 4
3
1
2, 4
TIME

FIG.40


Access Start
Data Transfer
Default State

FIG.41

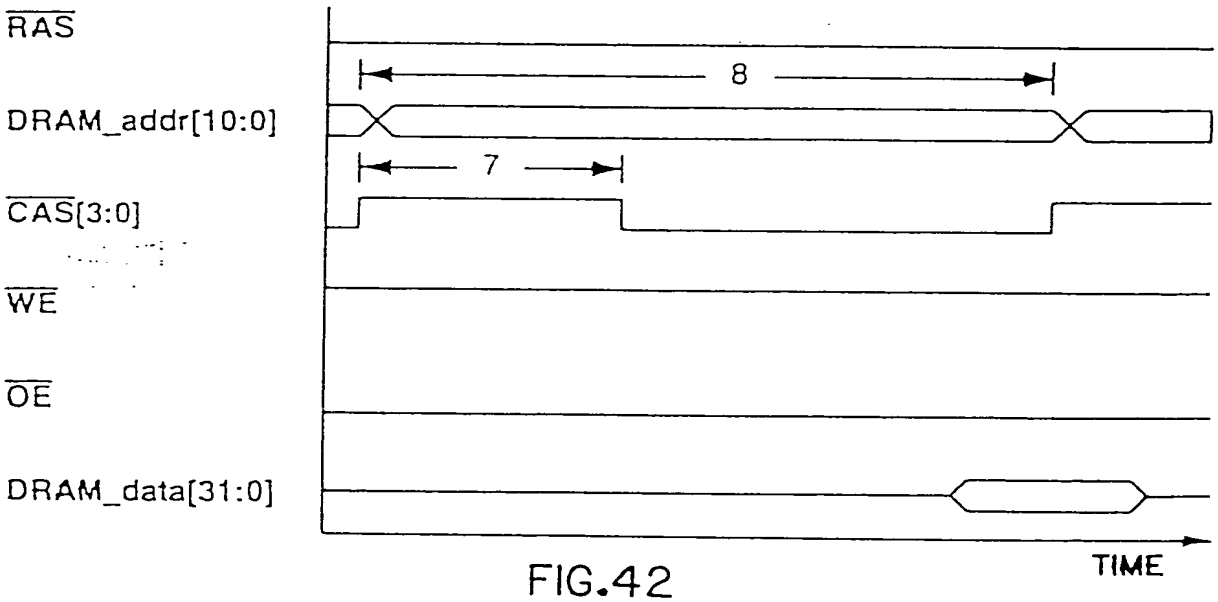

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
8
7
TIME

FIG.42

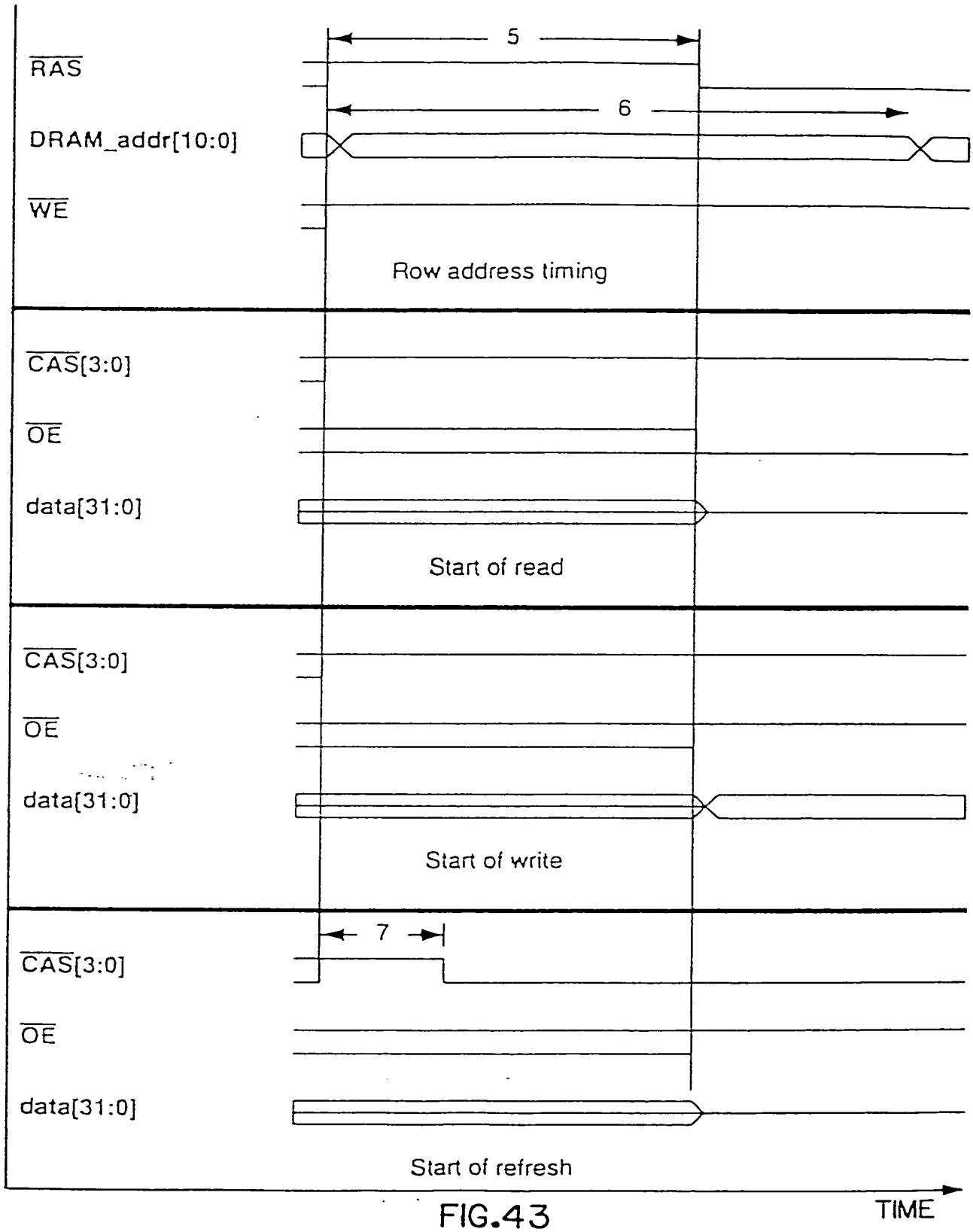

5
RAS
6
DRAM_addr[10:0]
WE
Row address timing
CAS[3:0]
OE
data[31:0]
Start of read
CAS[3:0]
OE
data[31:0]
Start of write
7
CAS[3:0]
OE
data[31:0]
Start of refresh
TIME

FIG.43

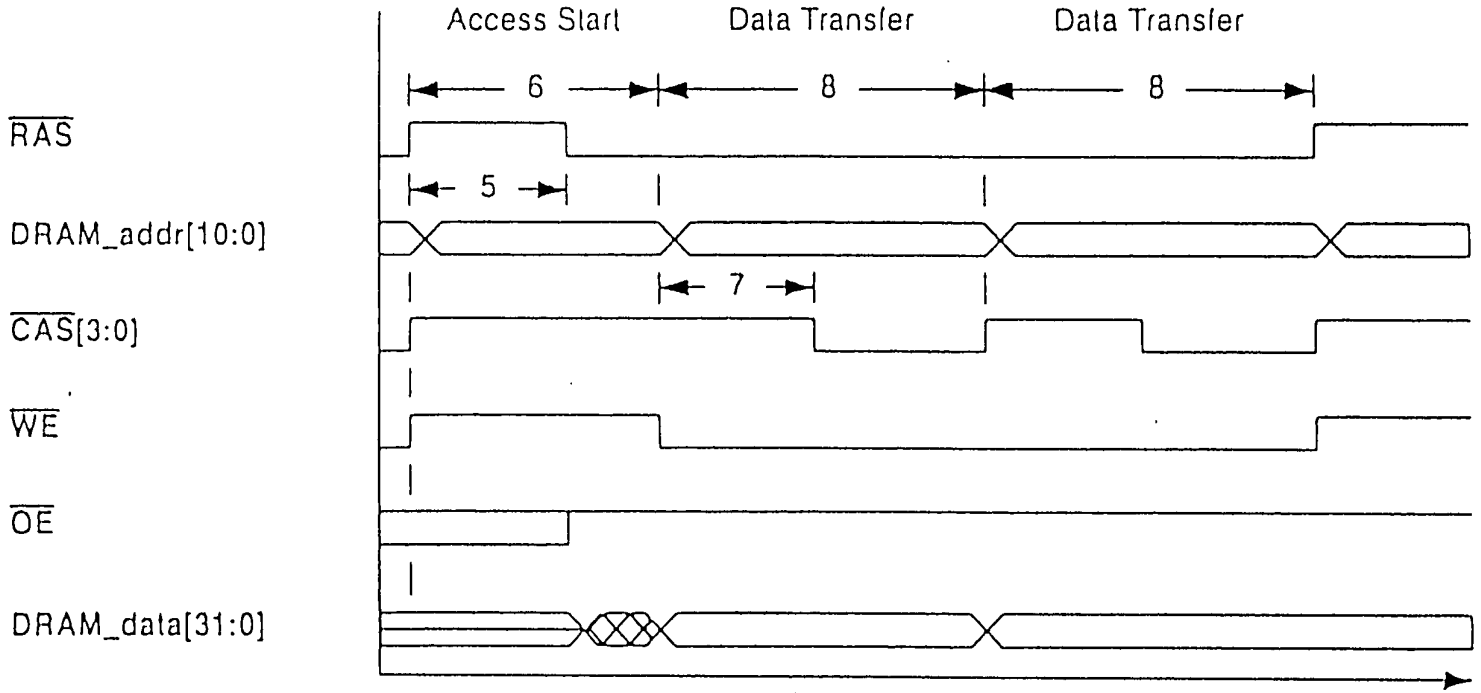


FIG.44

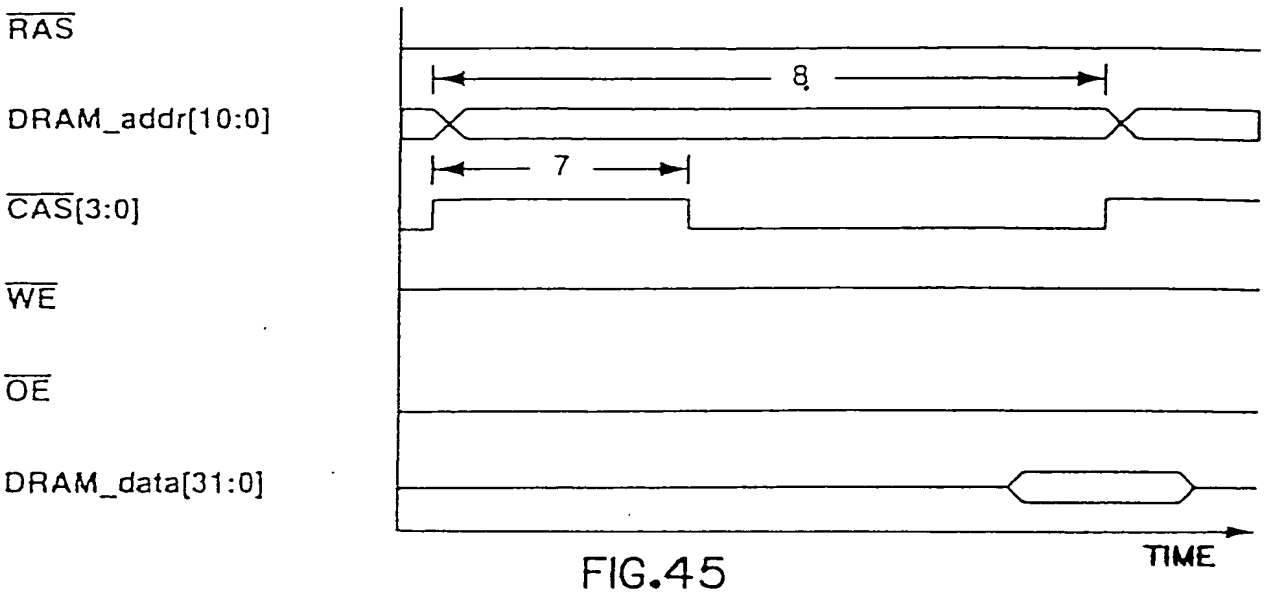

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
8
7
TIME

FIG.45

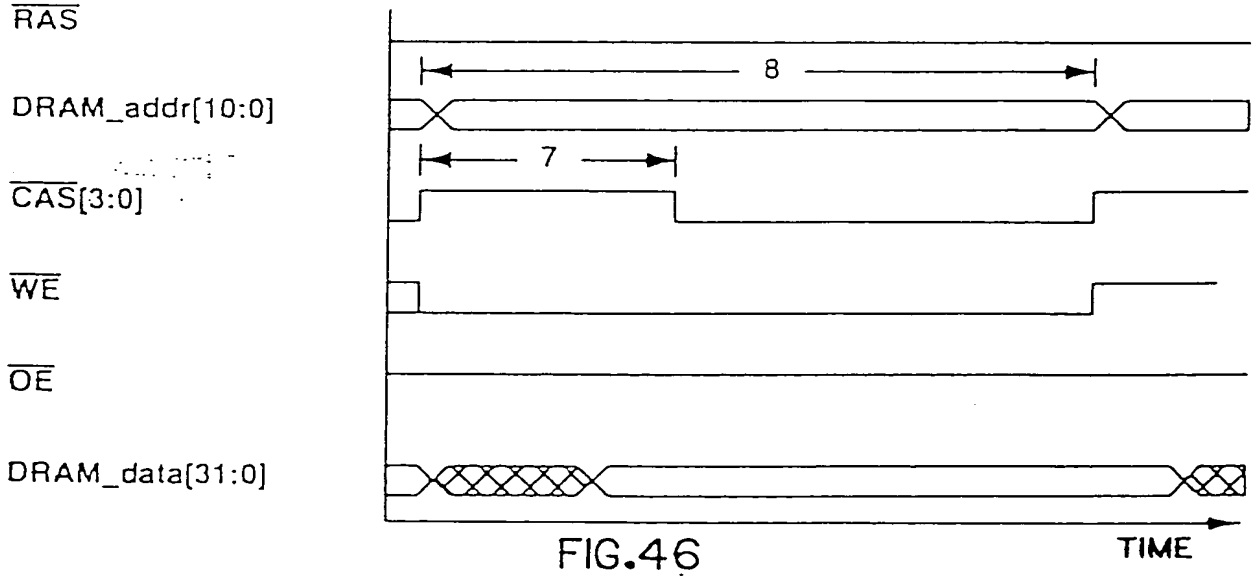

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
8
7
TIME

FIG.46

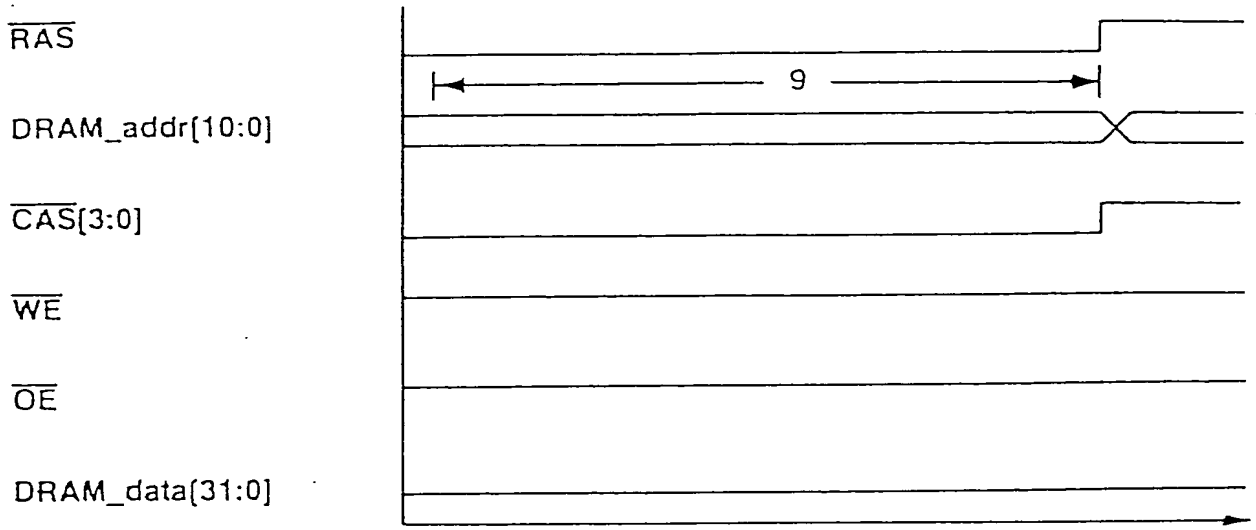

RAS
DRAM_addr[10:0]
CAS[3:0]
WE
OE
DRAM_data[31:0]
9
TIME

FIG.47

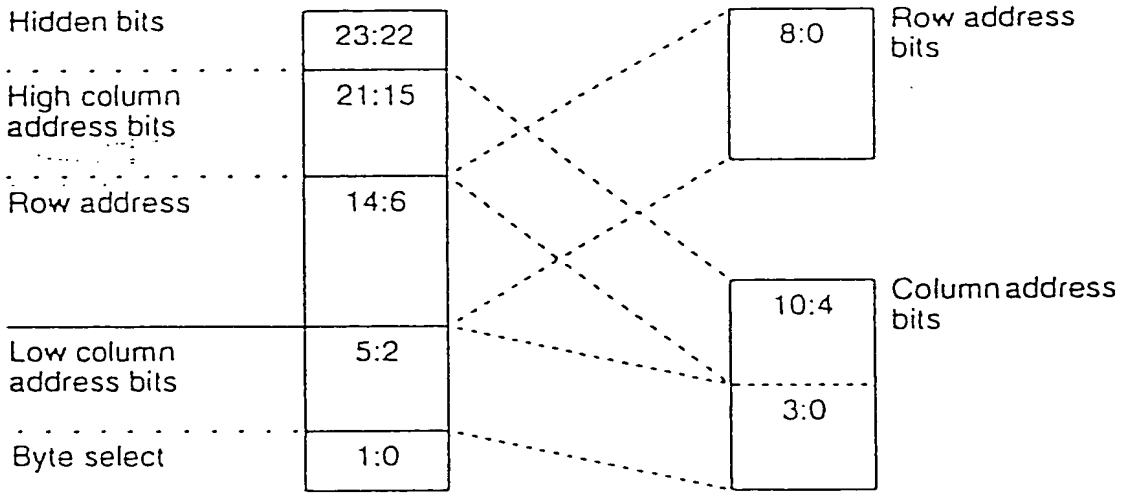

Hidden bits
High column address bits
Row address
Low column address bits
Byte select
23:22
21:15
14:6
5:2
1:0
8:0
Row address bits
10:4
3:0
Column address bits

FIG.48

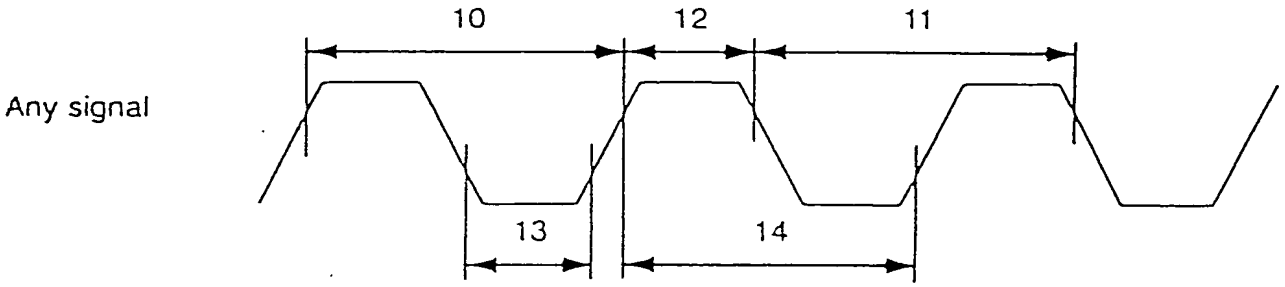

10
12
11
Any signal
13
14

FIG.49

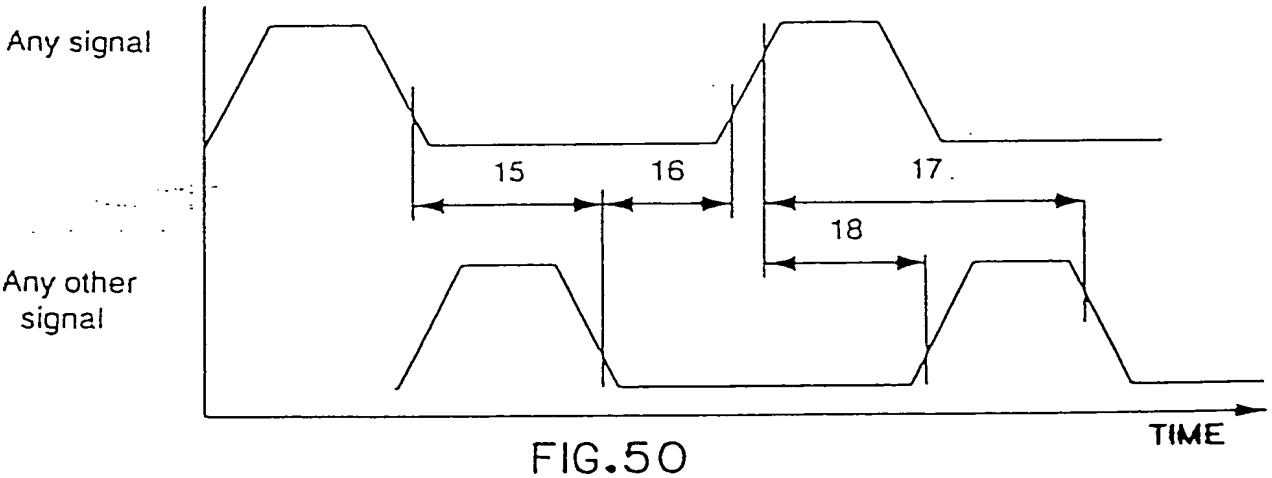

Any signal
15
16
17
18
Any other signal
TIME

FIG.50

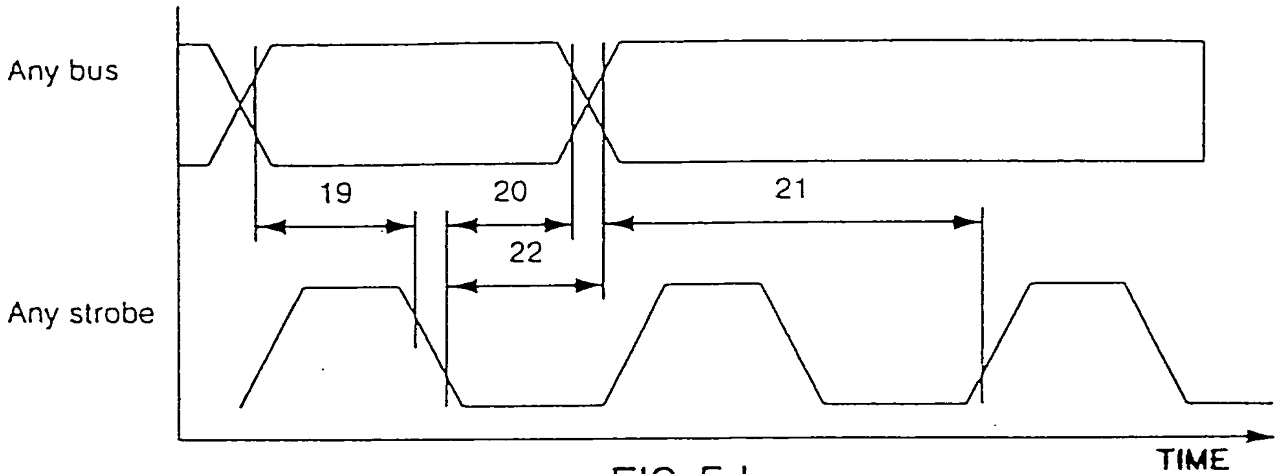


FIG.51

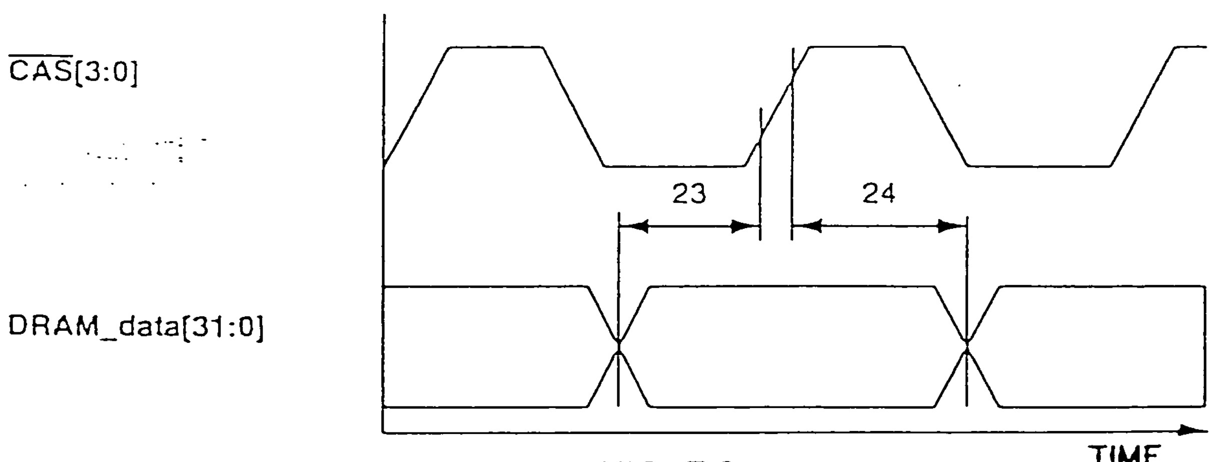


FIG.52

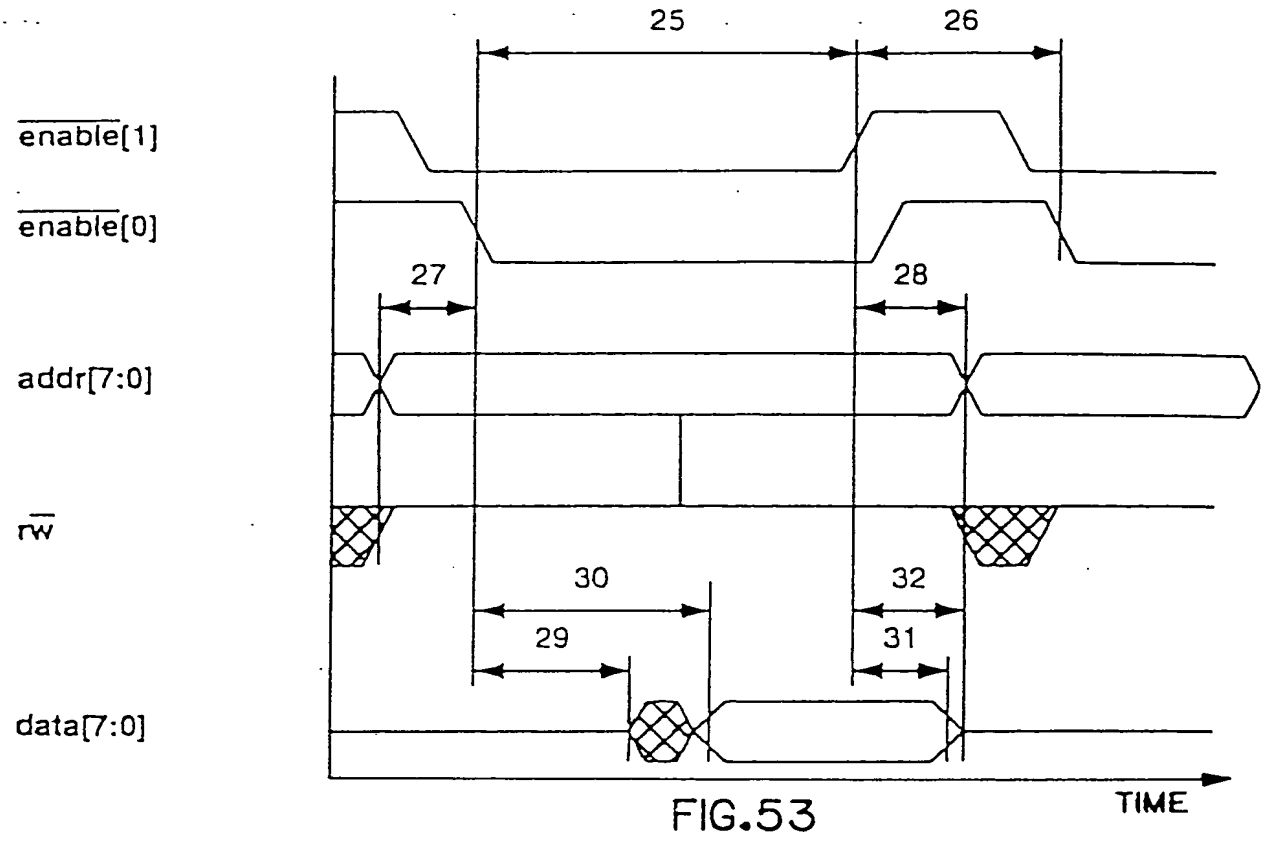

25
26
enable[1]
enable[0]
27
28
addr[7:0]
rw
30
32
29
31
data[7:0]
TIME

FIG.53

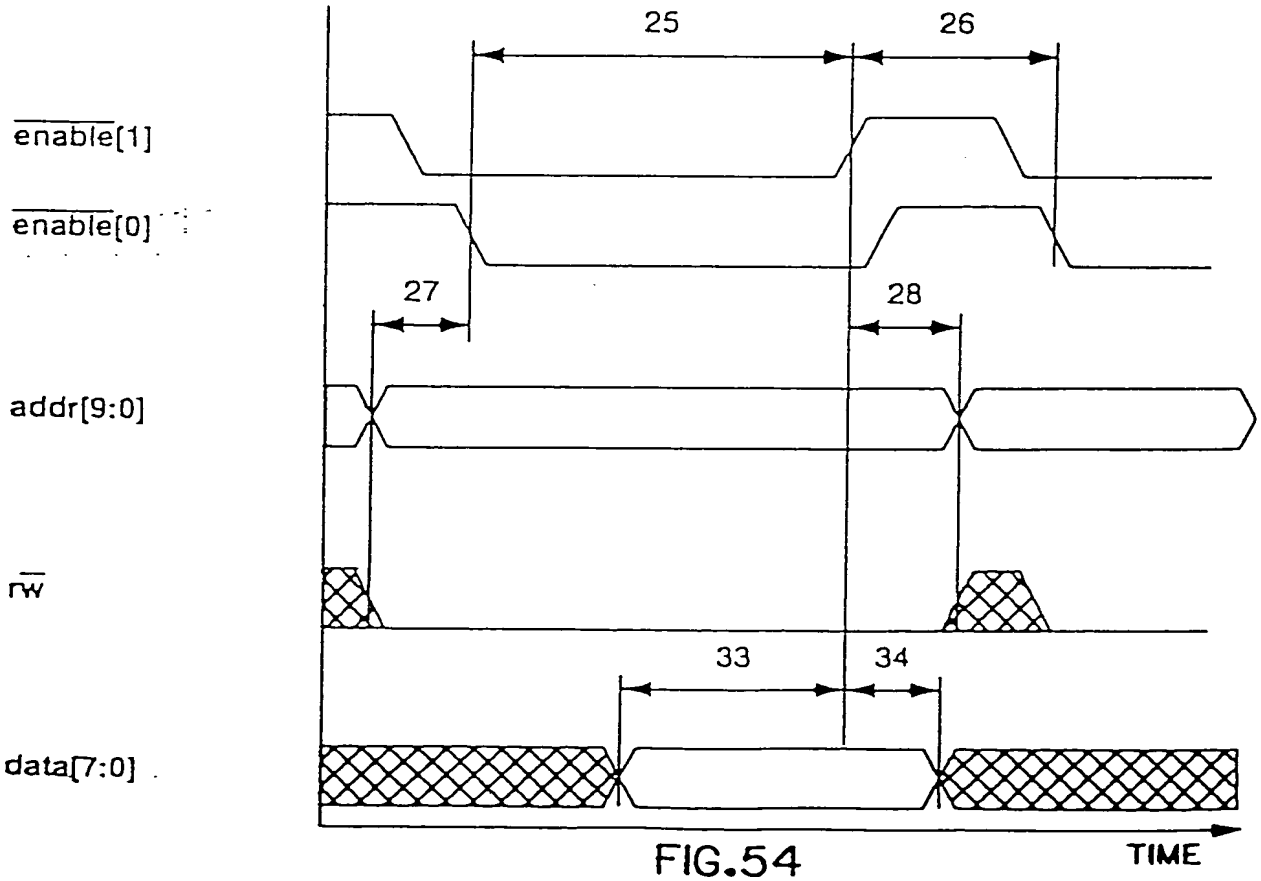

25
26
enable[1]
enable[0]
27
28
addr[9:0]
rw
33
34
data[7:0]
TIME

FIG.54

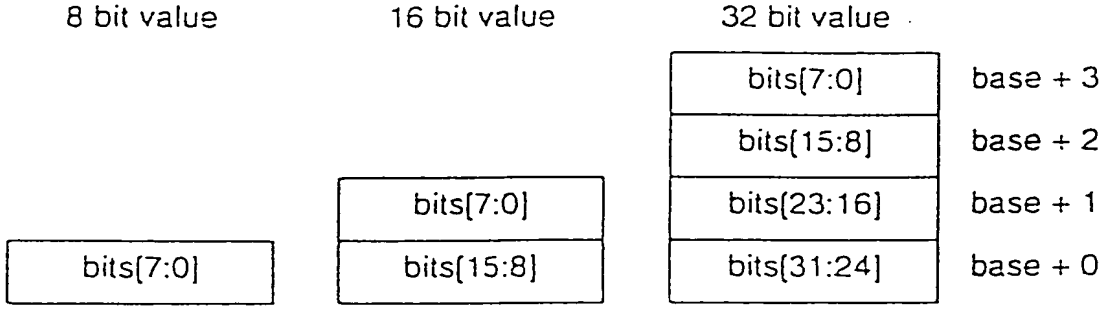


FIG.55

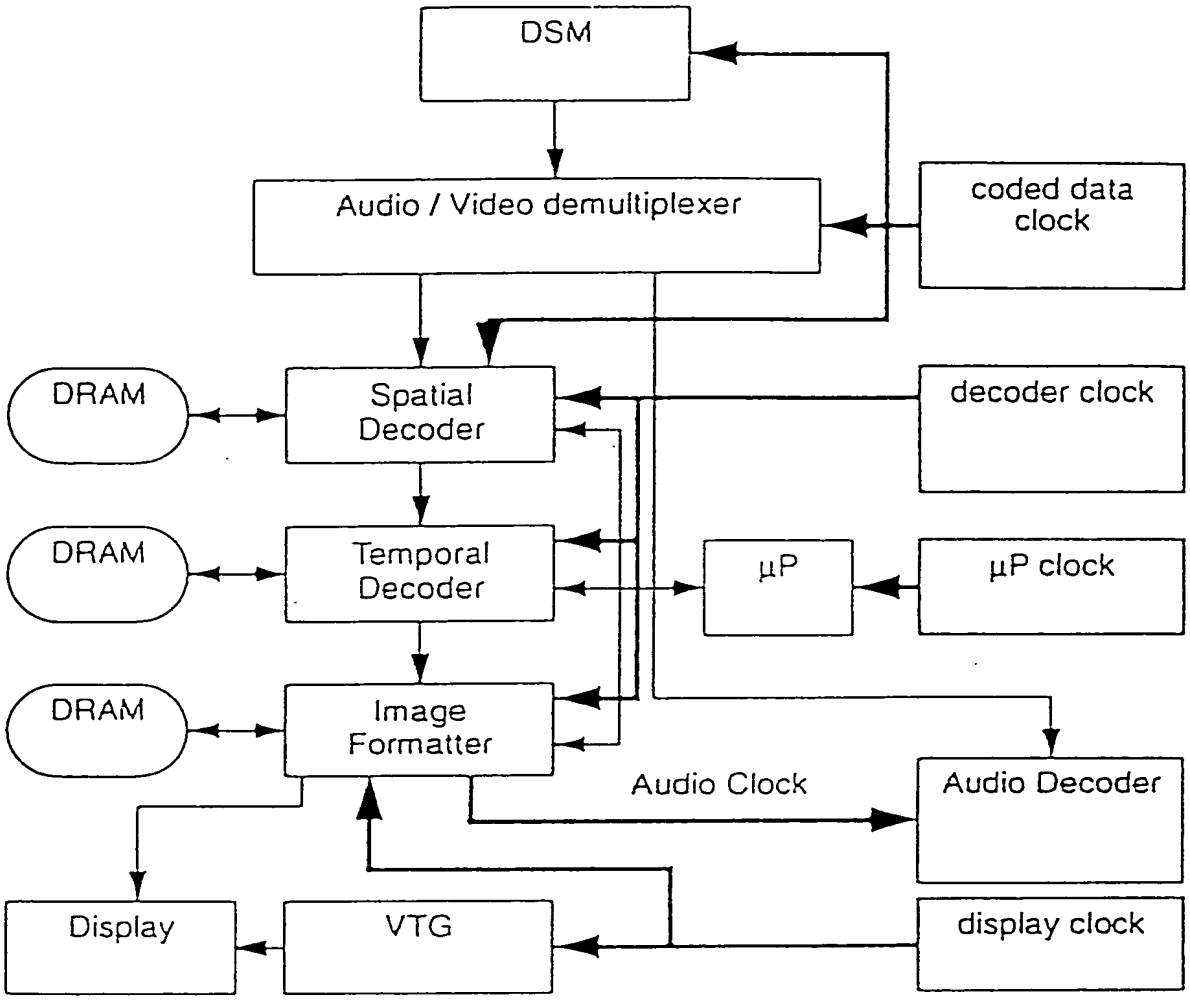


FIG.56

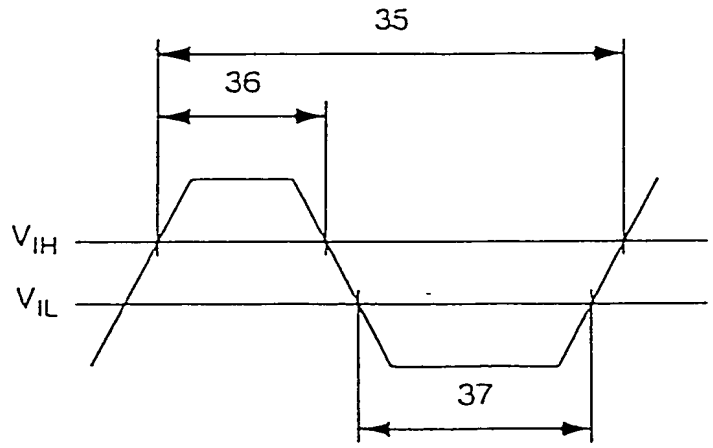


FIG.57

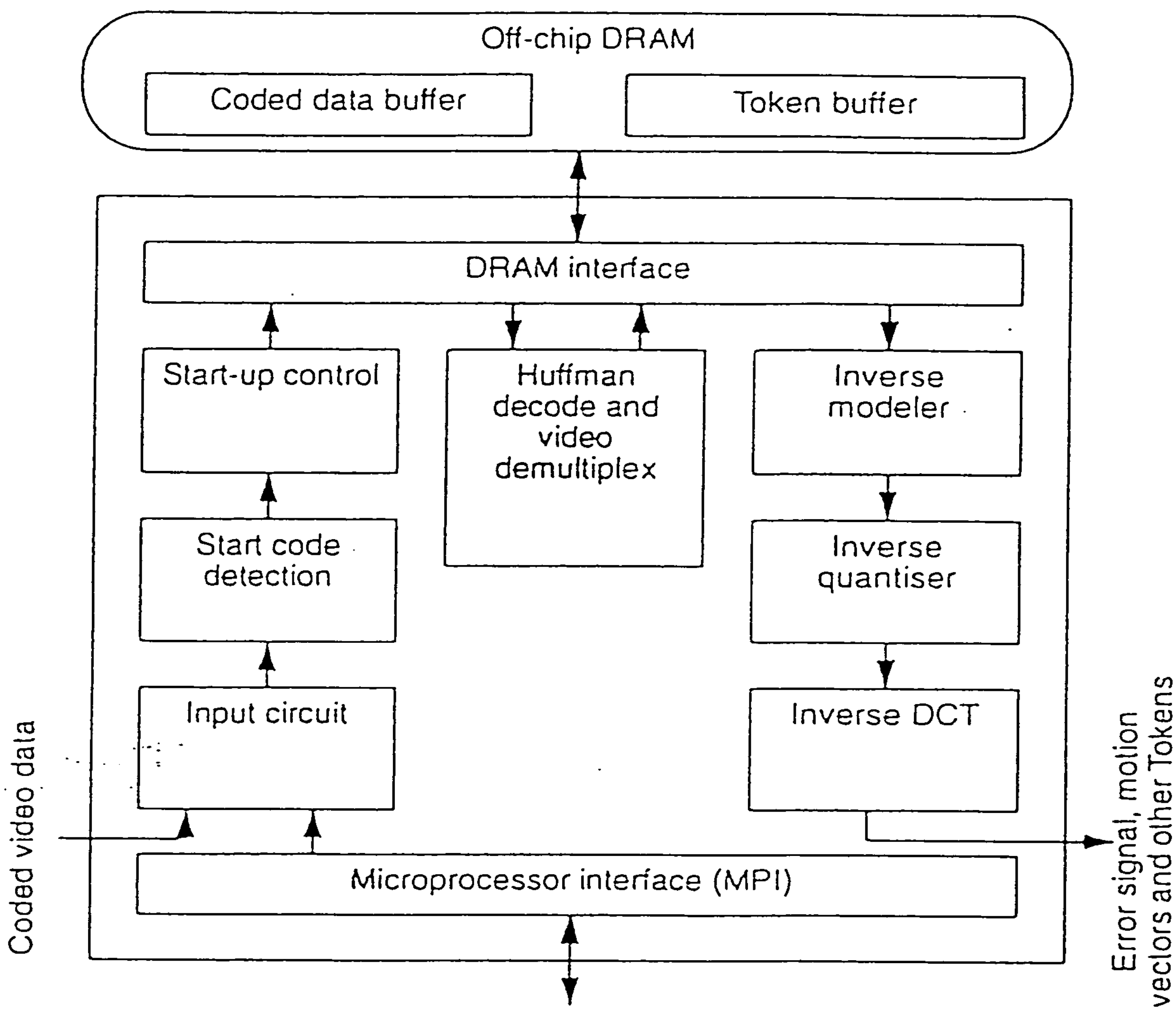

Off-chip DRAM
Coded data buffer
Token buffer
DRAM interface
Start-up control
Huffman decode and video demultiplex
Inverse modeler
Start code detection
Inverse quantiser
Input circuit
Inverse DCT
Coded video data
Microprocessor interface (MPI)
Error signal, motion vectors and other Tokens

FIG.58

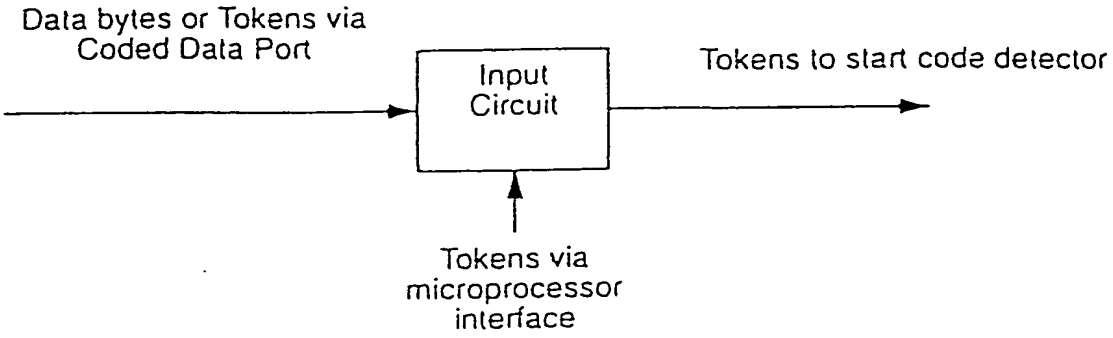

Data bytes or Tokens via
Coded Data Port
Input
Circuit
Tokens to start code detector
Tokens via
microprocessor
interface

FIG.59

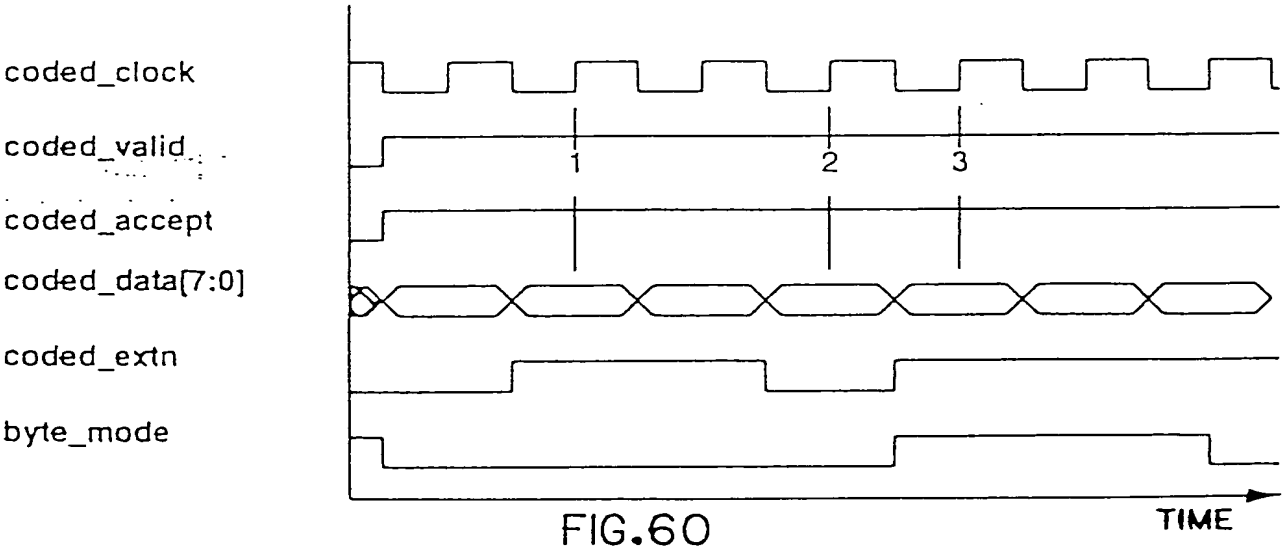

coded_clock
coded_valid
coded_accept
coded_data[7:0]
coded_extn
byte_mode
1
2
3
TIME

FIG.60

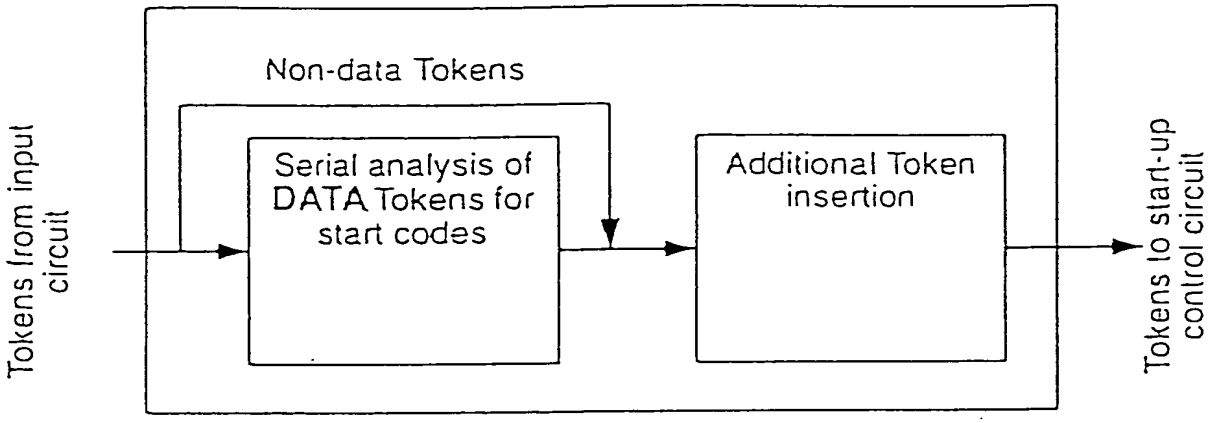


FIG.61

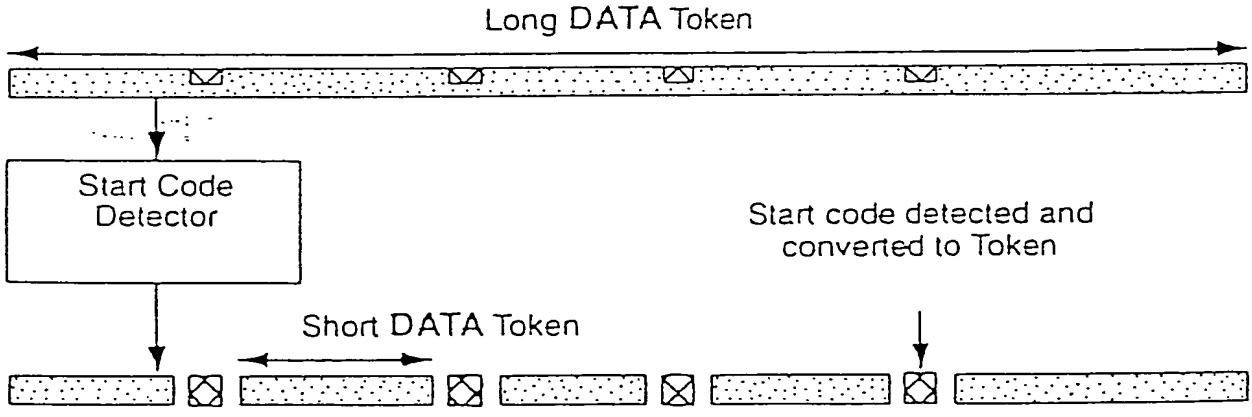


FIG.62

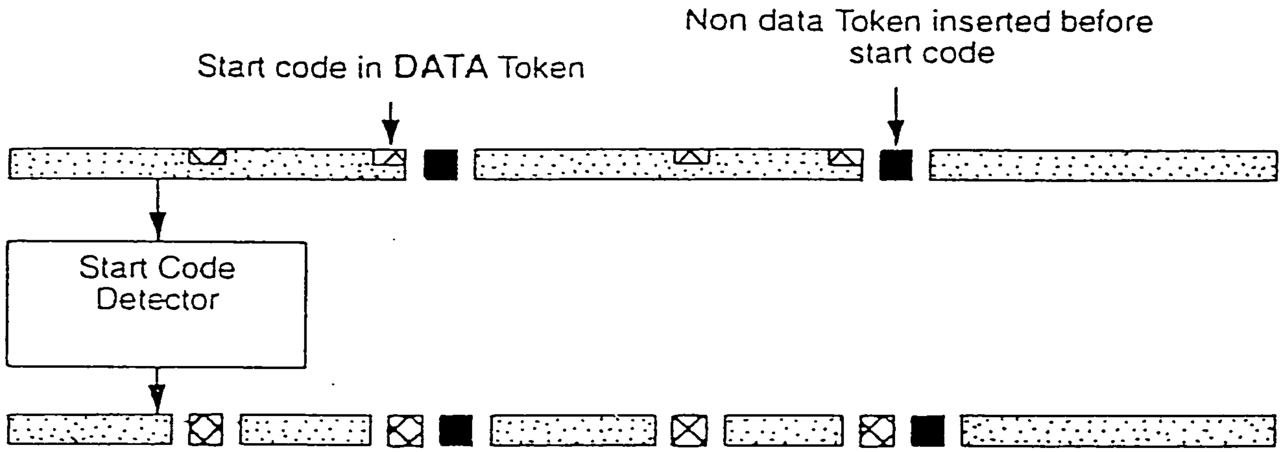

Non data Token inserted before start code
Start code in DATA Token
Start Code Detector

FIG.63

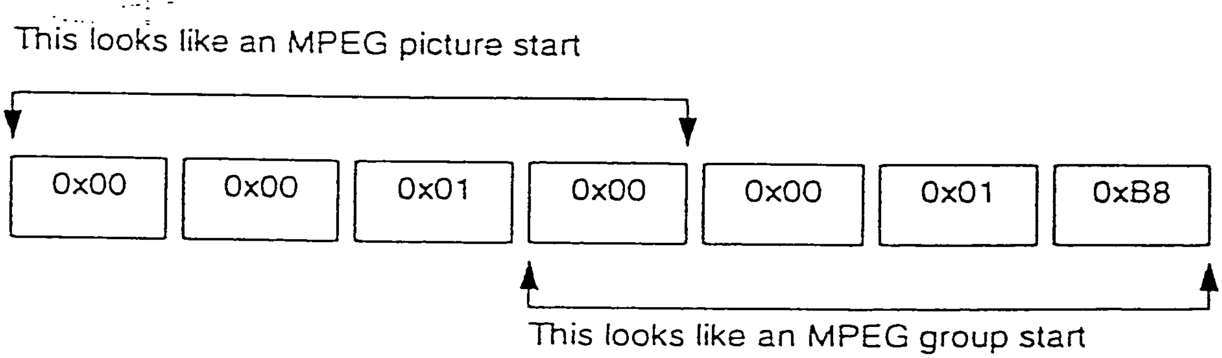

This looks like an MPEG picture start
0x00
0x00
0x01
0x00
0x00
0x01
0xB8
This looks like an MPEG group start

FIG.64

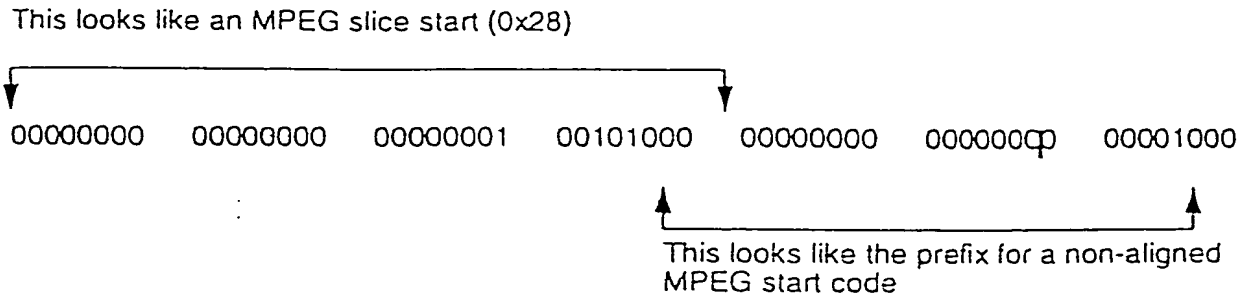

This looks like an MPEG slice start (0x28)
00000000 00000000 00000001 00101000 00000000 00000000 00001000
This looks like the prefix for a non-aligned MPEG start code

FIG.65

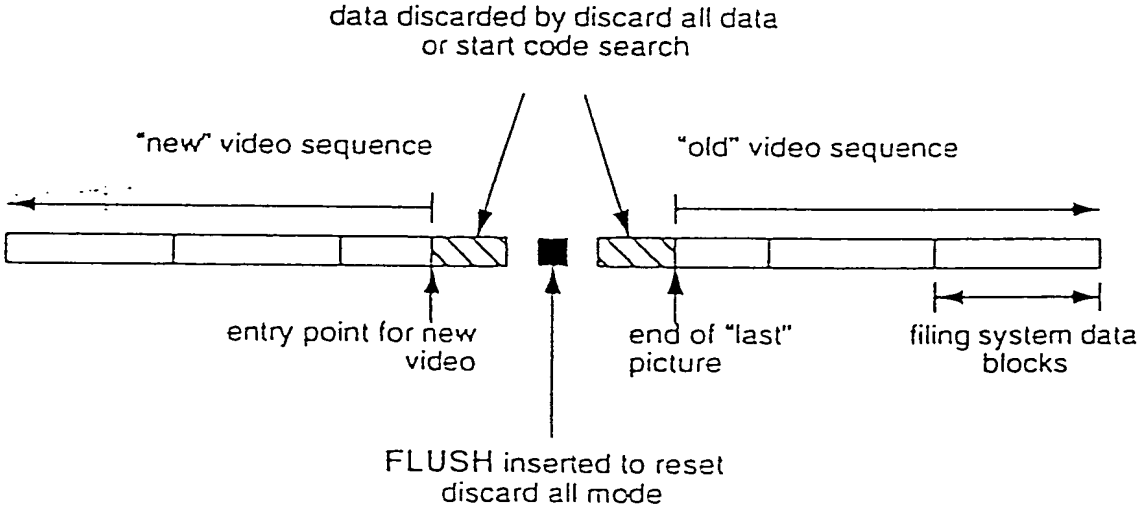

data discarded by discard all data or start code search
"new" video sequence
"old" video sequence
entry point for new video
end of "last" picture
filing system data blocks
FLUSH inserted to reset discard all mode

FIG.66

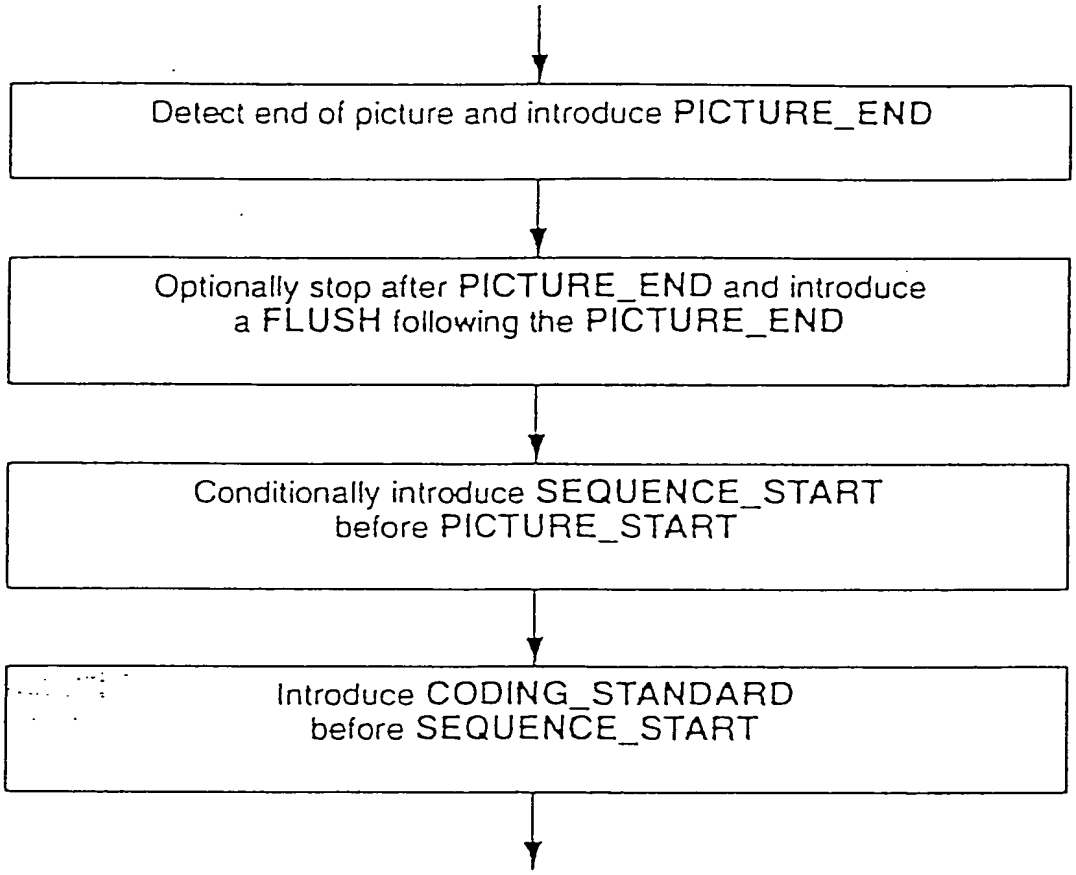


FIG.67

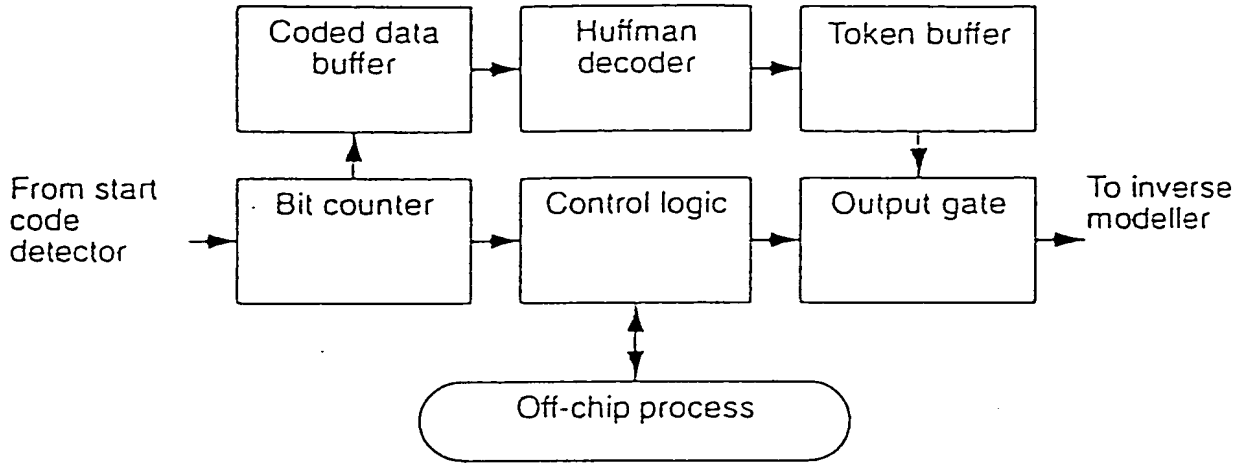

Coded data buffer
Huffman decoder
Token buffer
From start code detector
Bit counter
Control logic
Output gate
To inverse modeller
Off-chip process

FIG.68

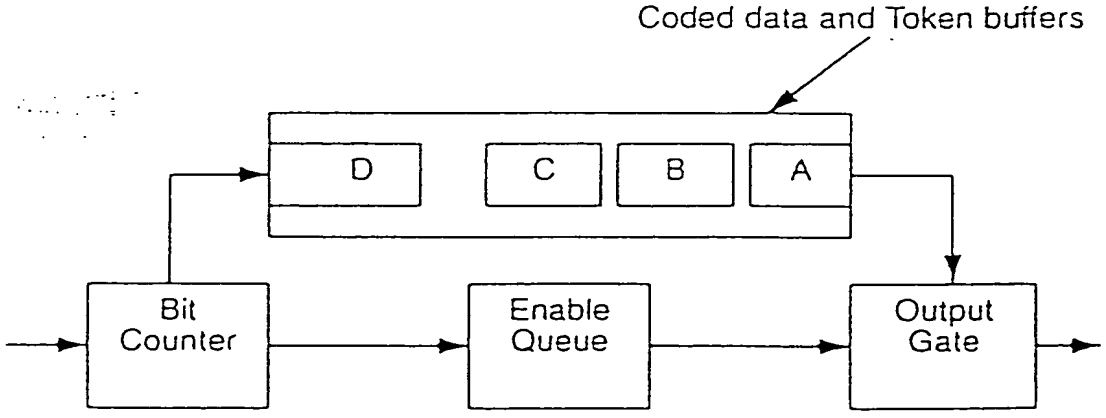

Coded data and Token buffers
D
C
B
A
Bit Counter
Enable Queue
Output Gate

FIG.69

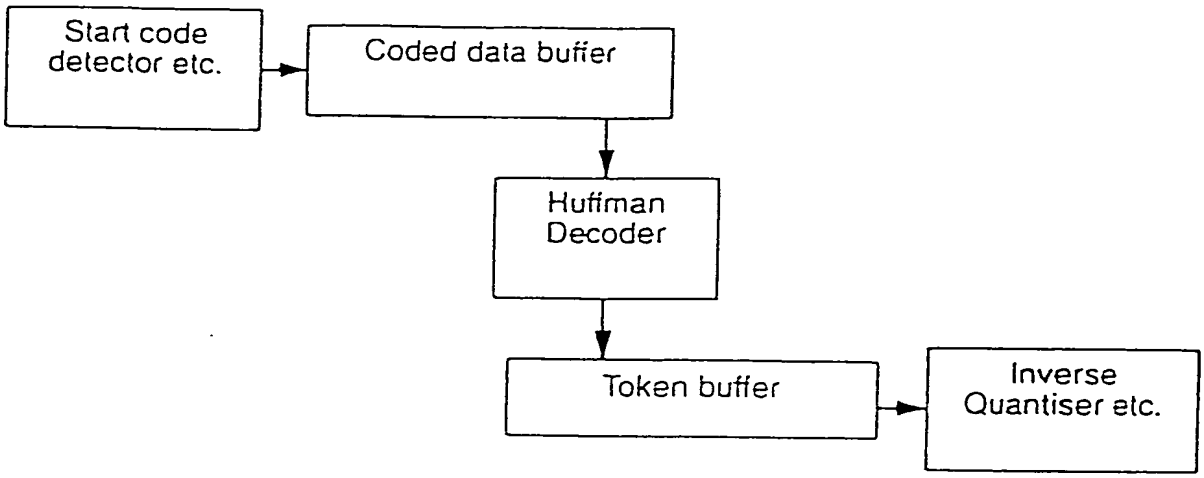

Start code detector etc.
Coded data buffer
Huffman Decoder
Token buffer
Inverse Quantiser etc.

FIG.70

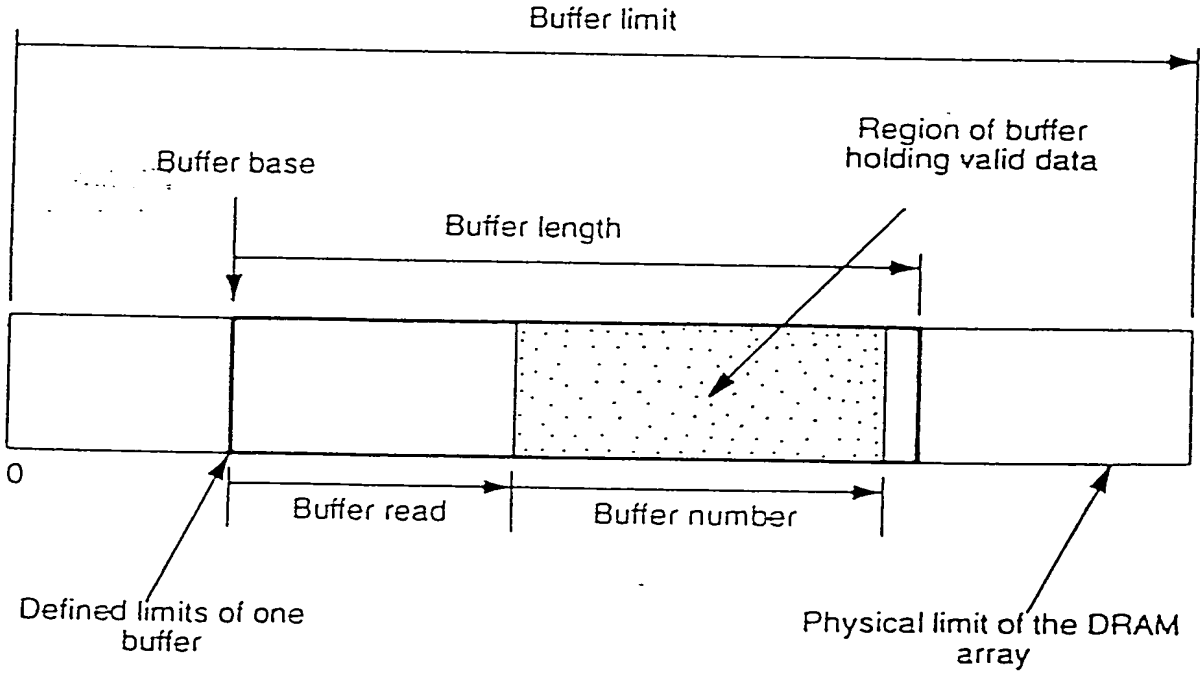

Buffer limit
Region of buffer holding valid data
Buffer base
Buffer length
0
Buffer read
Buffer number
Defined limits of one buffer
Physical limit of the DRAM array

FIG.71

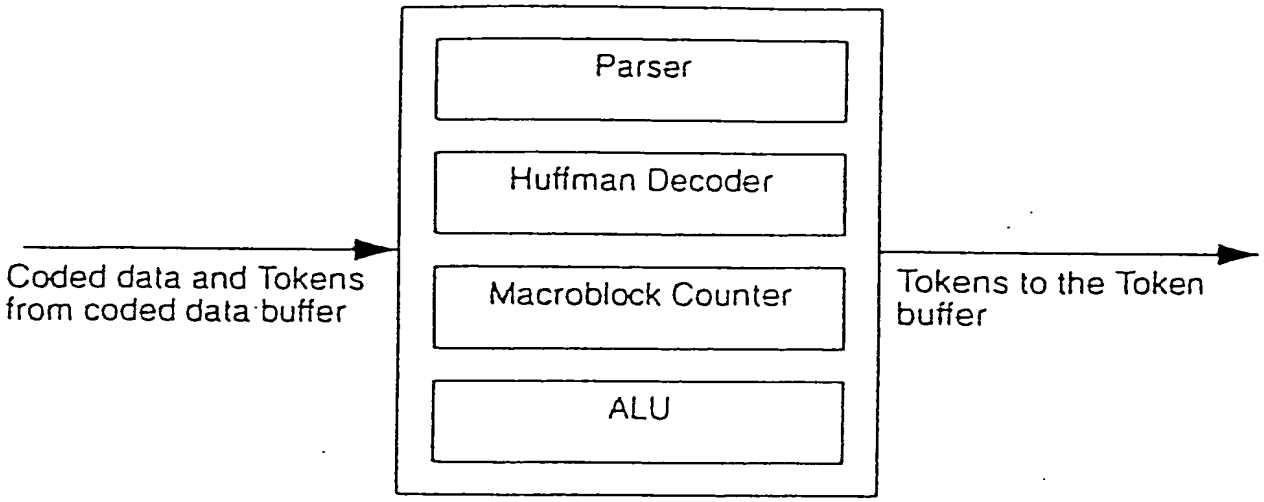

Parser
Huffman Decoder
Macroblock Counter
ALU
Coded data and Tokens from coded data buffer
Tokens to the Token buffer

FIG.72

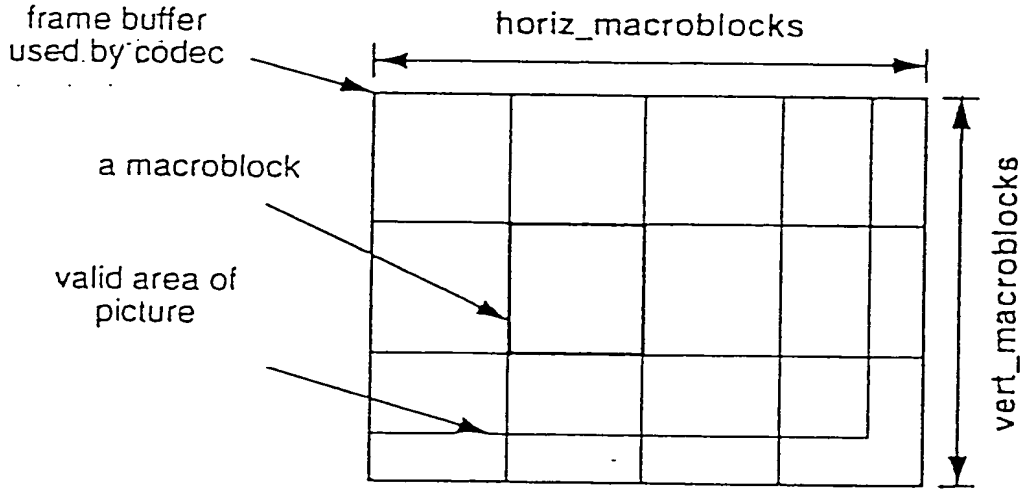

frame buffer used by codec
horiz_macroblocks
a macroblock
valid area of picture
vert_macroblocks

FIG.73

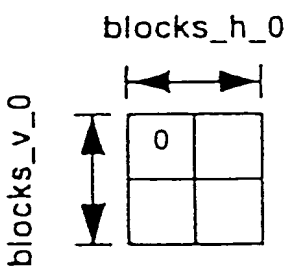


FIG.74A

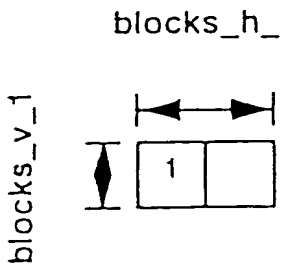


FIG.74B

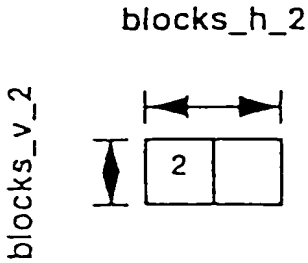


FIG.74C

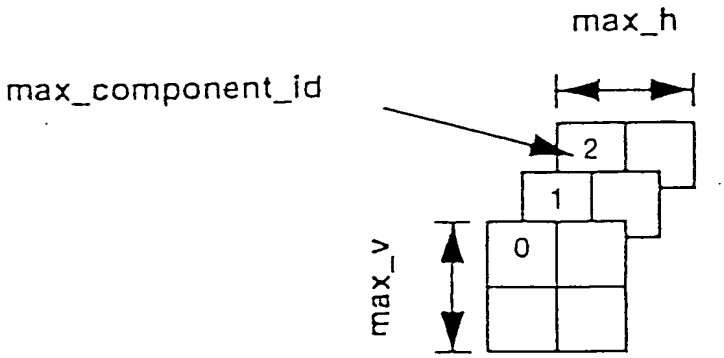


FIG.74D

$$\begin{cases} \text{horiz\_macroblocks} = \dfrac{\text{horiz\_pels} + 15}{16} \\ \text{vert\_macroblocks} = \dfrac{\text{vert\_pels} + 15}{16} \end{cases}$$

FIG.75

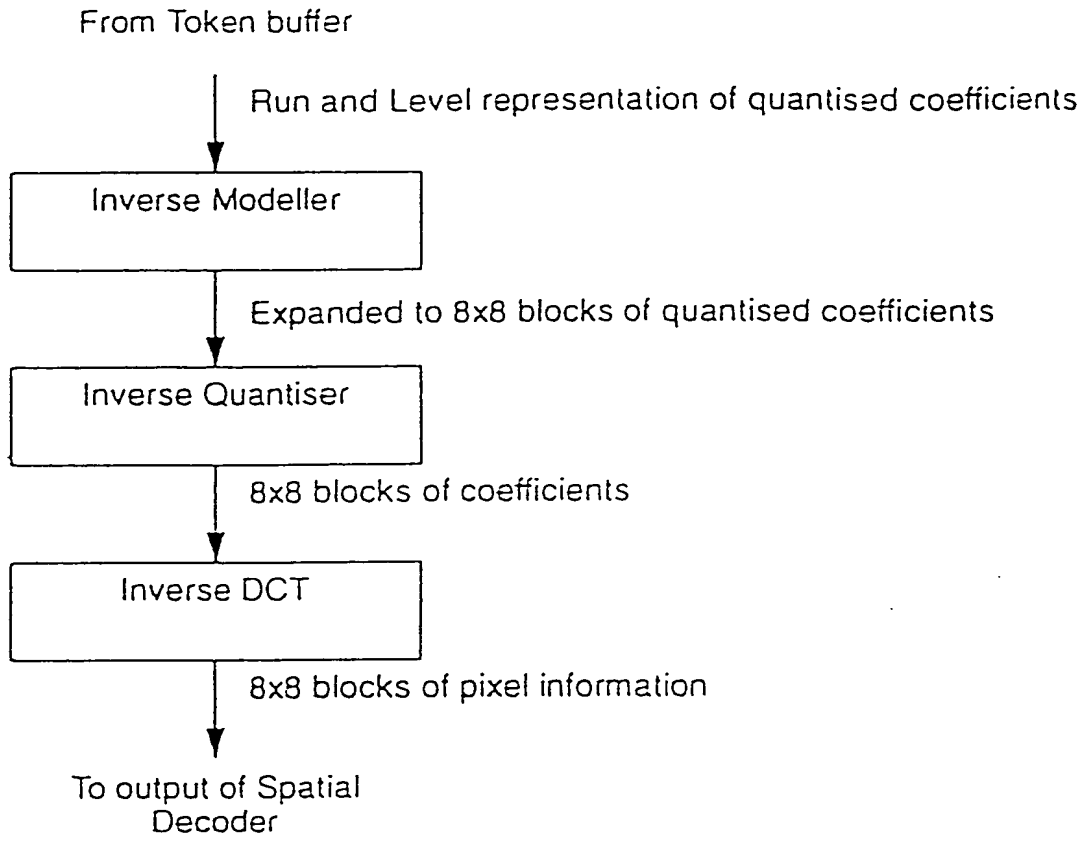

From Token buffer
Run and Level representation of quantised coefficients
Inverse Modeller
Expanded to 8x8 blocks of quantised coefficients
Inverse Quantiser
8x8 blocks of coefficients
Inverse DCT
8x8 blocks of pixel information
To output of Spatial Decoder

FIG.76

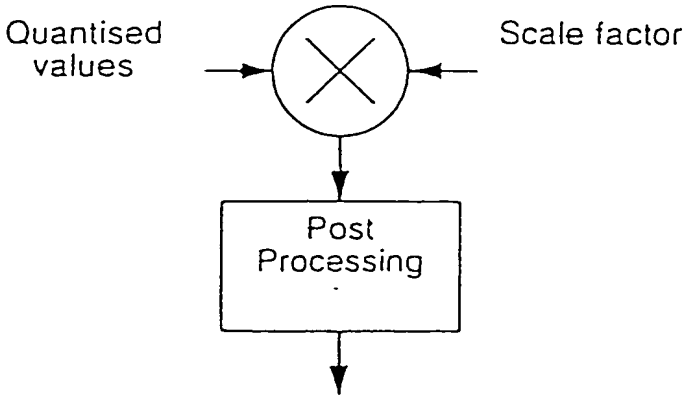

Quantised values
Scale factor
Post Processing

FIG.77

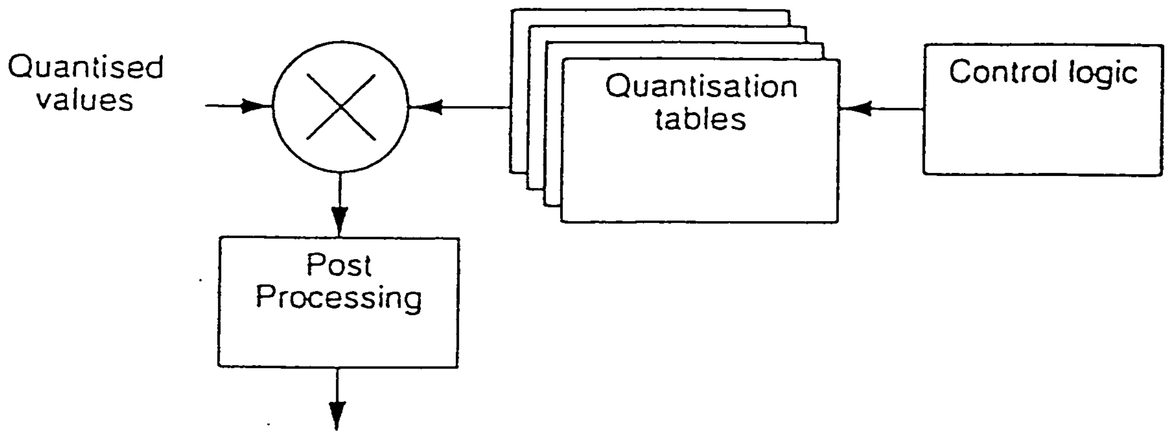


FIG.78

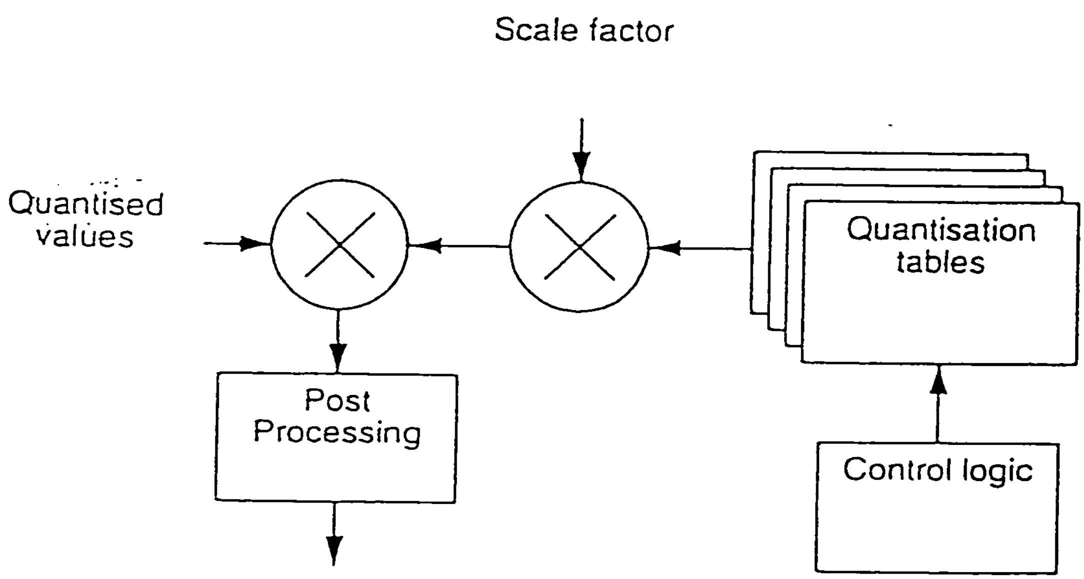


FIG.79

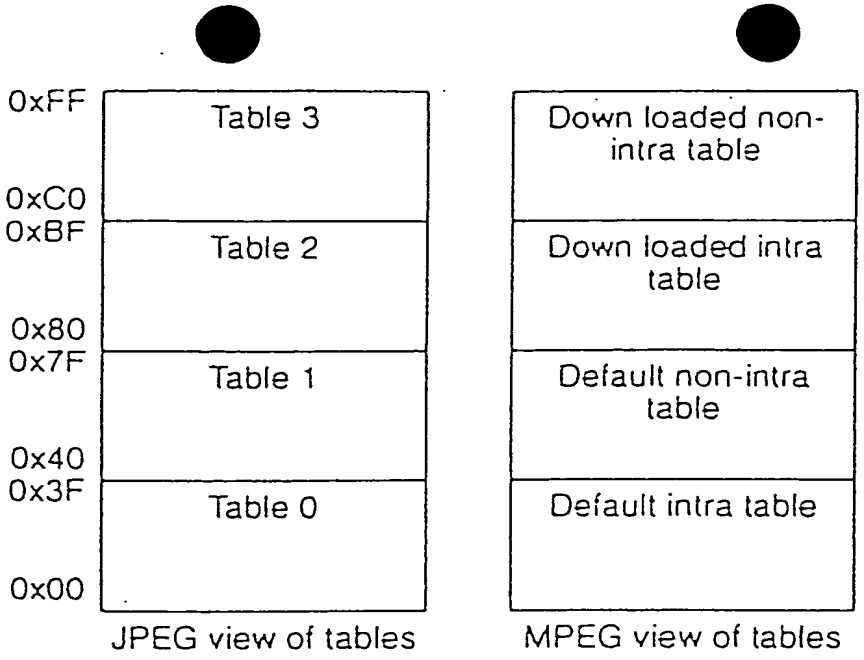


FIG.80

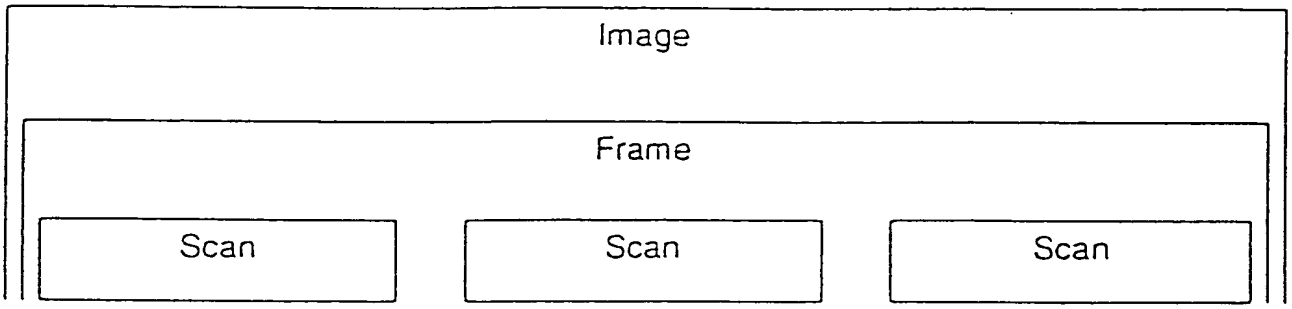


FIG.81

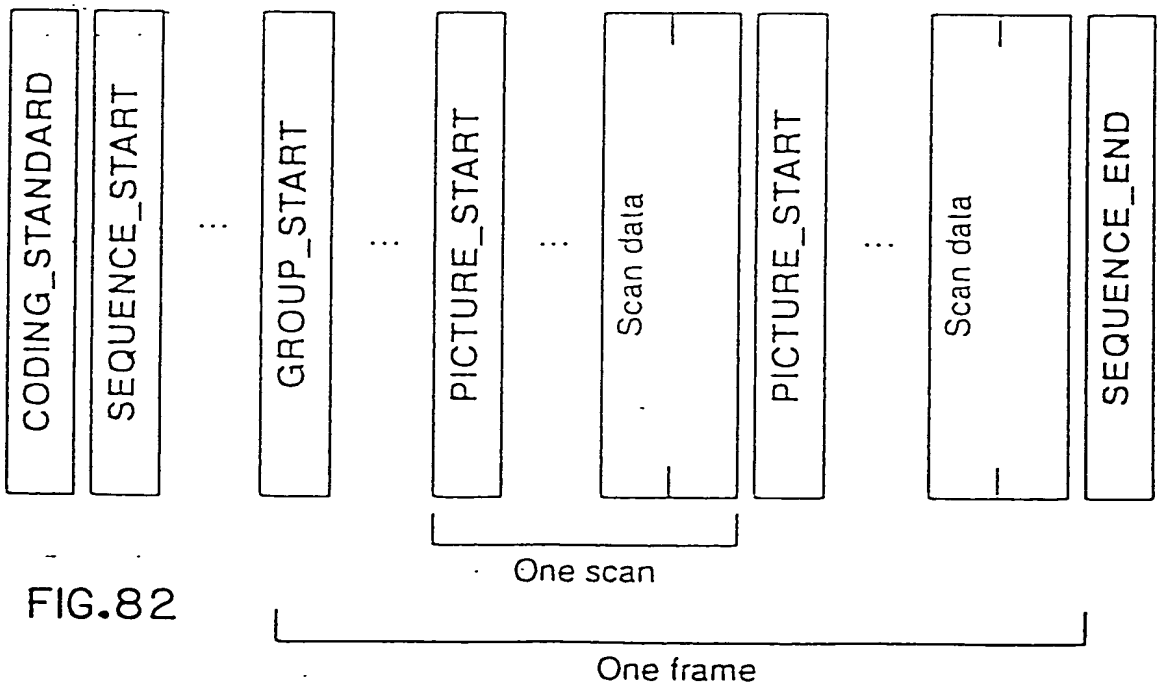


FIG.82

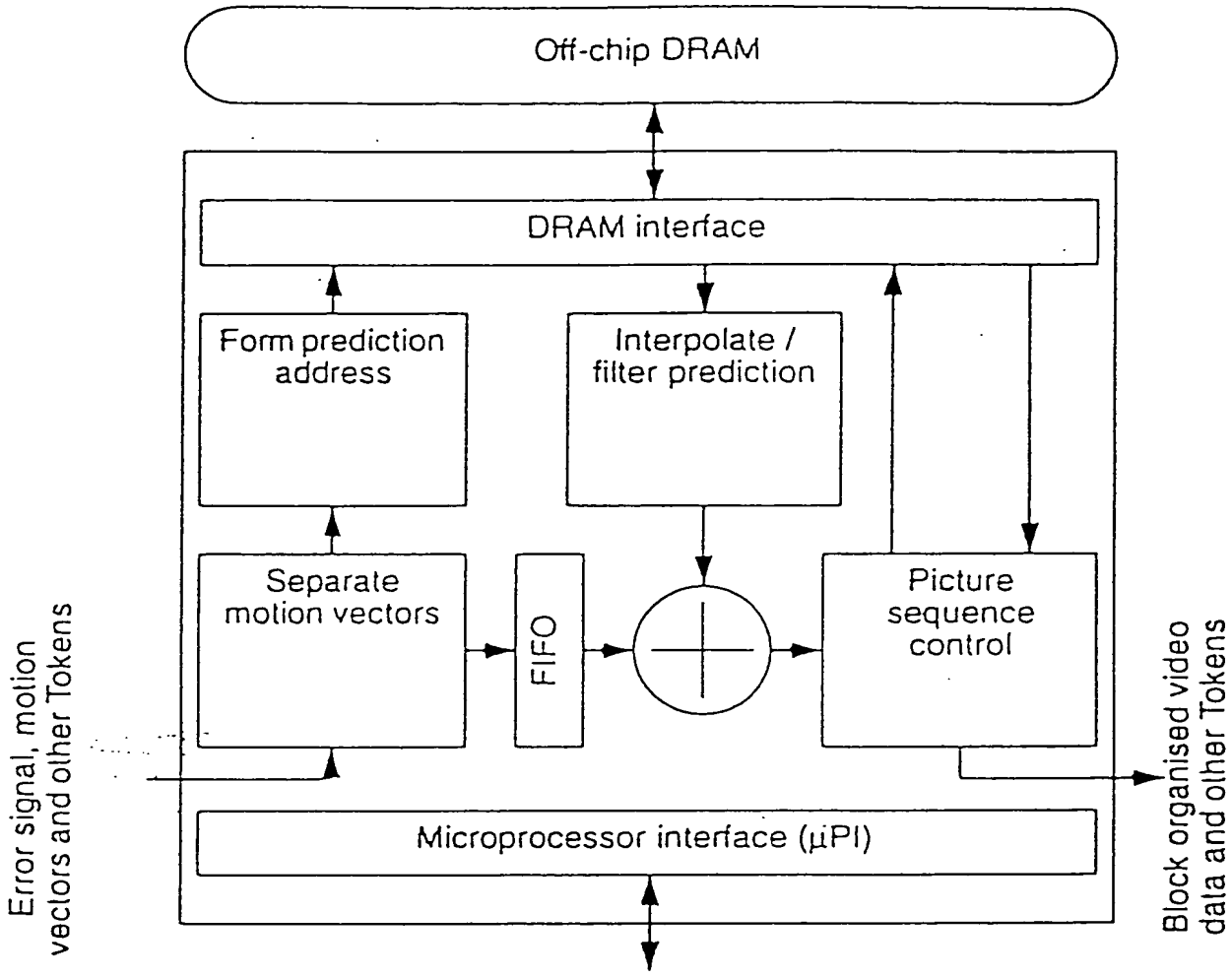


FIG.83

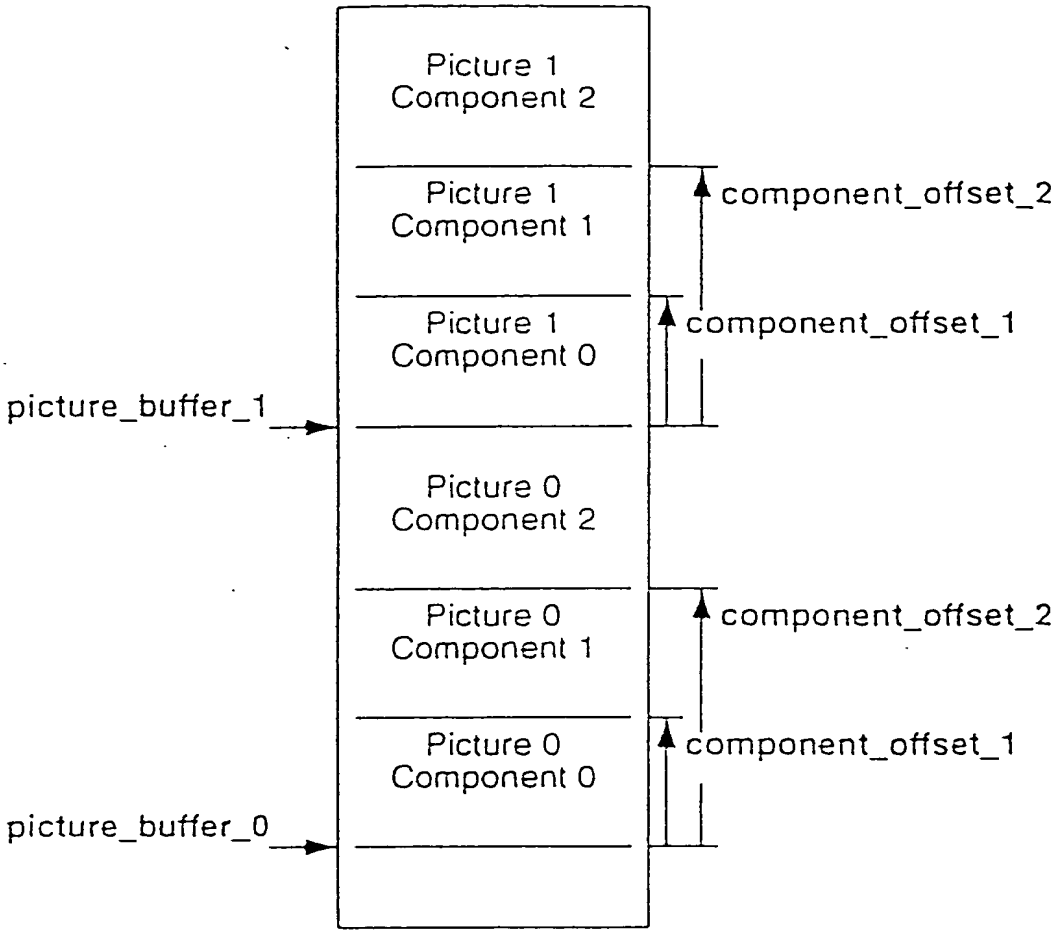

Picture 1
Component 2
Picture 1
Component 1
Picture 1
Component 0
Picture 0
Component 2
Picture 0
Component 1
Picture 0
Component 0
component_offset_2
component_offset_1
component_offset_2
component_offset_1
picture_buffer_1
picture_buffer_0

FIG.84

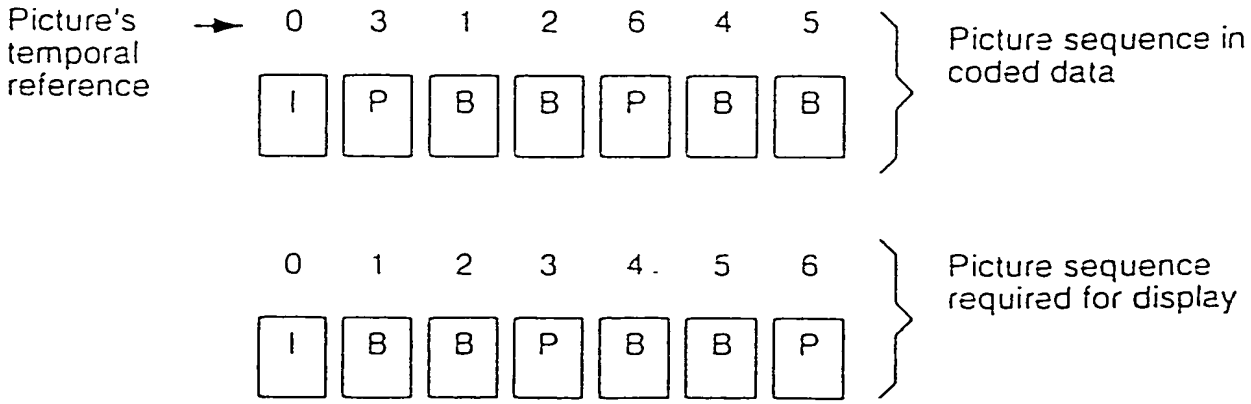

Picture's
temporal
reference
0 3 1 2 6 4 5
I P B B P B B
Picture sequence in
coded data
0 1 2 3 4 5 6
I B B P B B P
Picture sequence
required for display

FIG.85

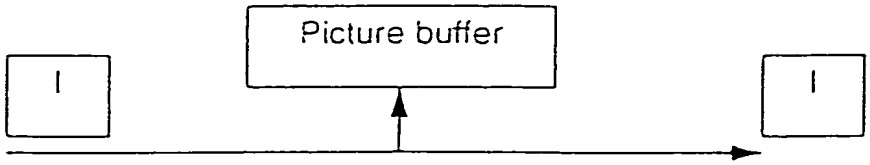


FIG.86

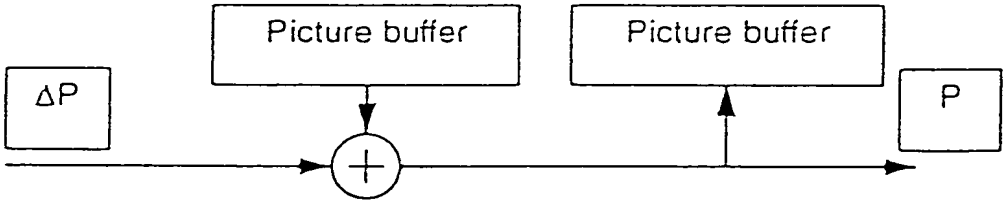


FIG.87

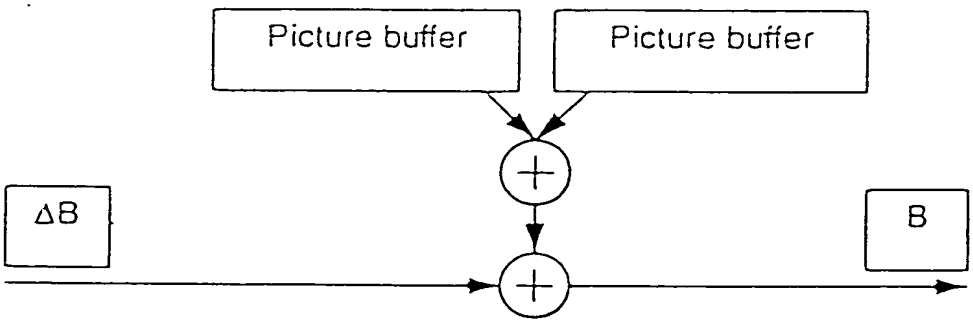


FIG.88

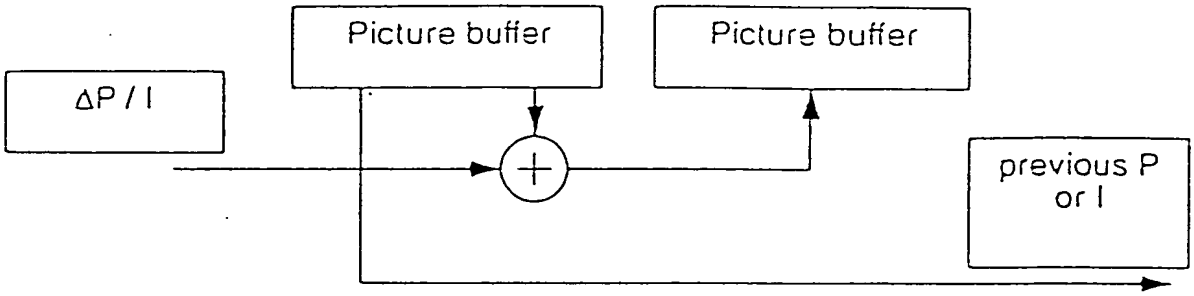


FIG.89

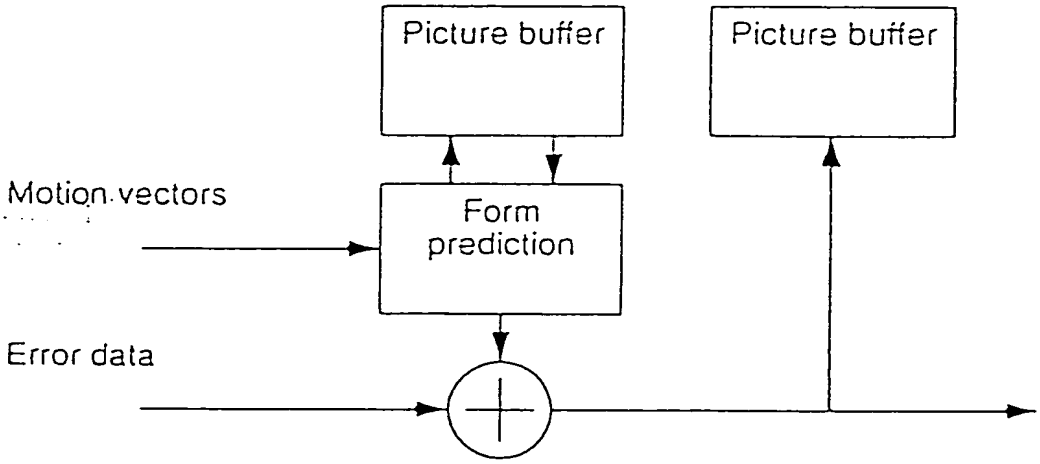


FIG.90

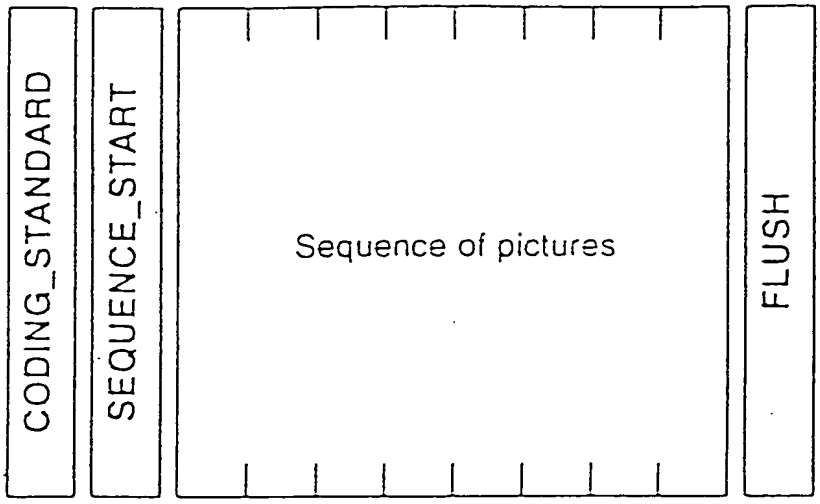


FIG.91

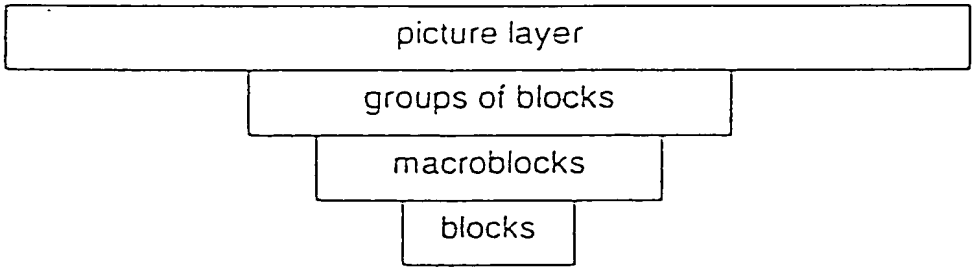


FIG.92

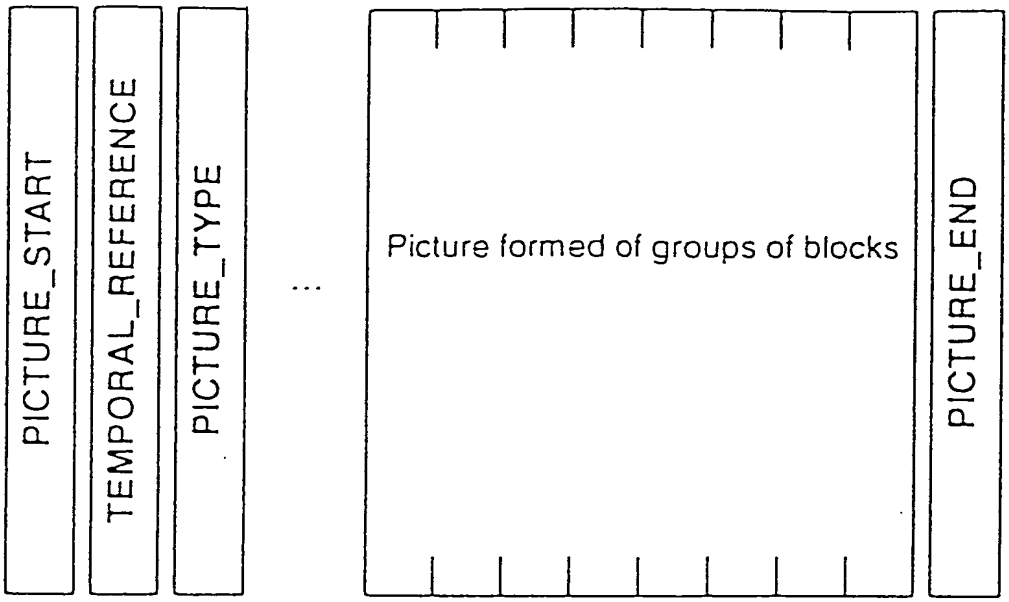

PICTURE_START
TEMPORAL_REFERENCE
PICTURE_TYPE
...
Picture formed of groups of blocks
PICTURE_END

FIG.93

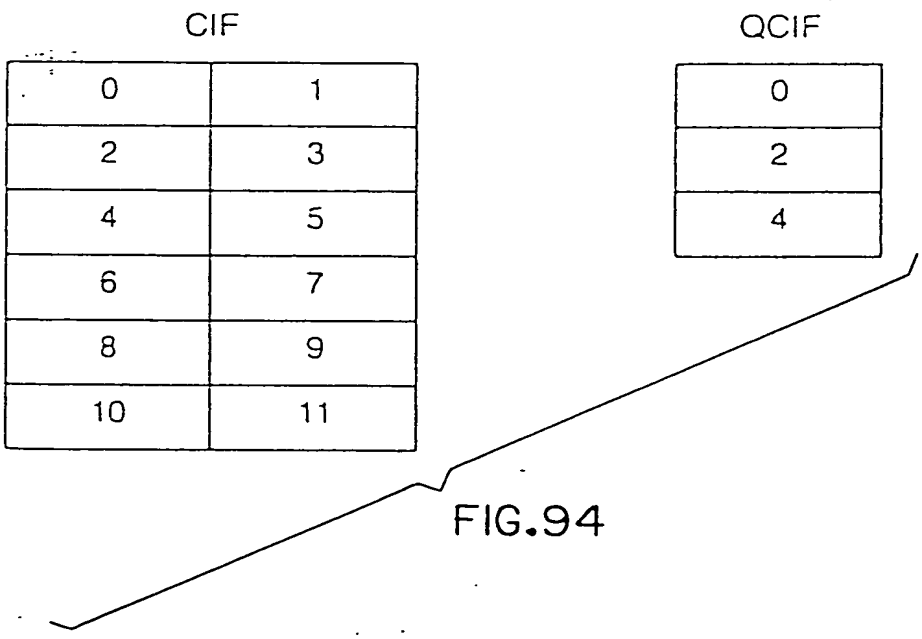

CIF
0
1
2
3
4
5
6
7
8
9
10
11
QCIF
0
2
4

FIG.94

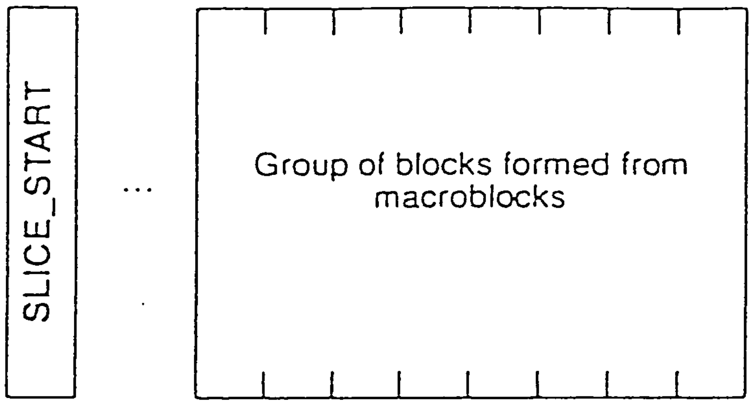


FIG.95

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33

FIG.96

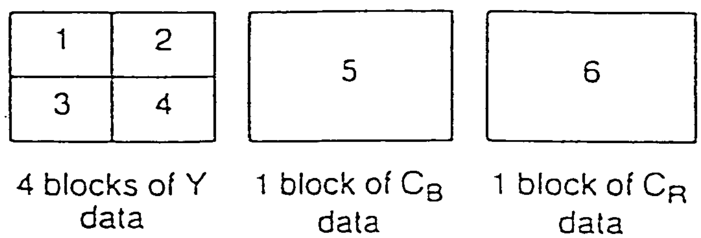

1
2
3
4
5
6
4 blocks of Y data
1 block of C_B data
1 block of C_R data

FIG.97

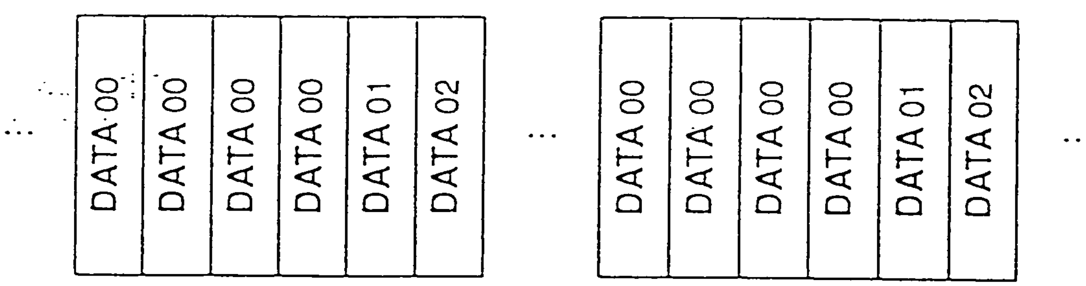

DATA 00
DATA 00
DATA 00
DATA 00
DATA 01
DATA 02
...
DATA 00
DATA 00
DATA 00
DATA 00
DATA 01
DATA 02
...

FIG.98

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

⋮

59	58	59	60	61	62	63	64

FIG.99

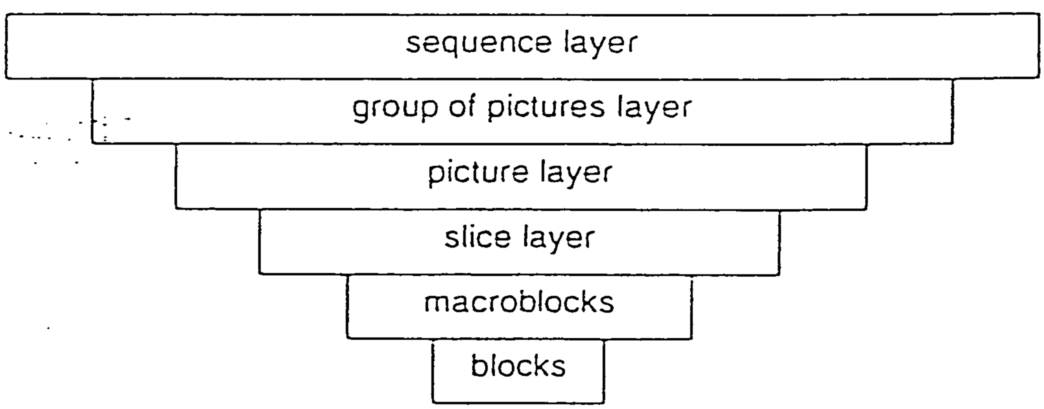


FIG.100

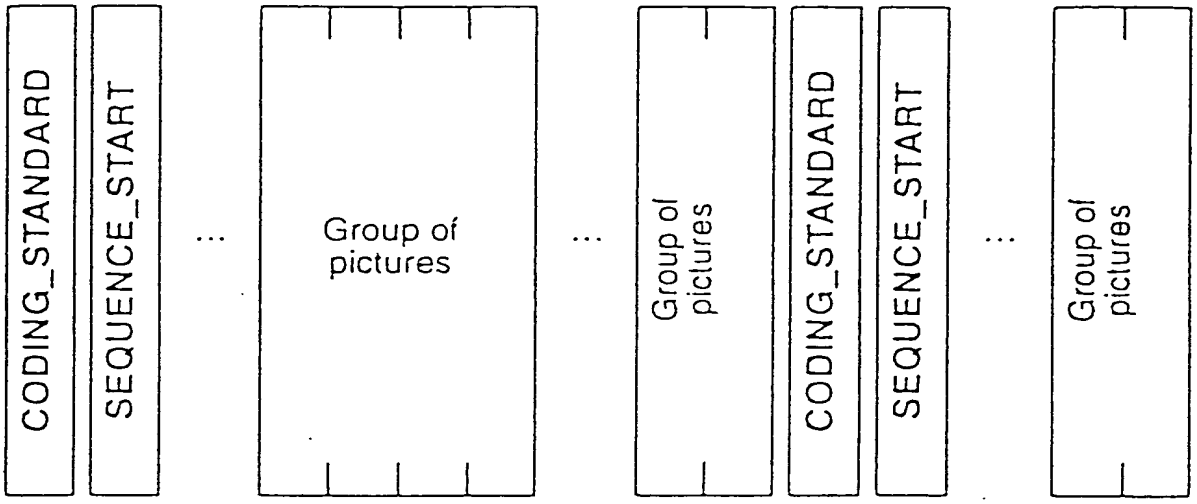


FIG. 101

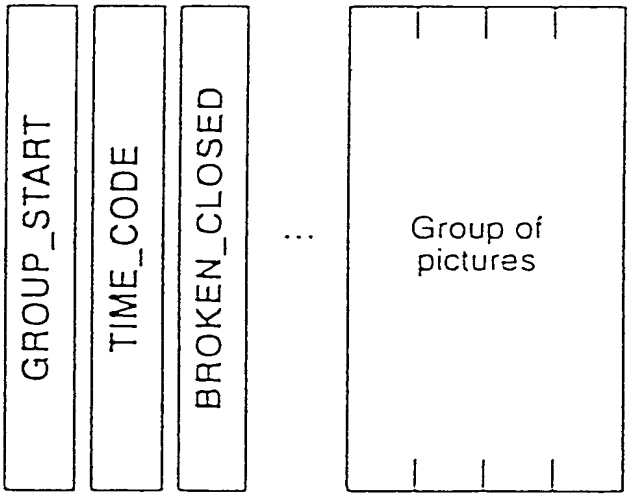


FIG. 102

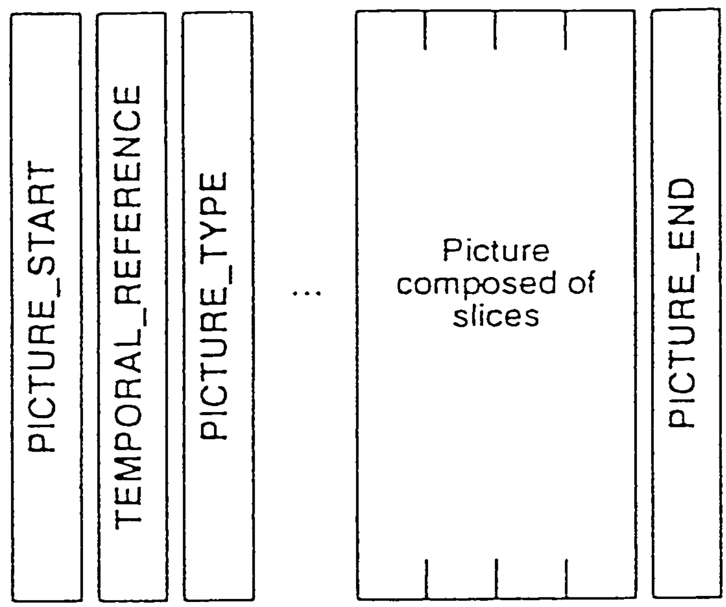


FIG. 103

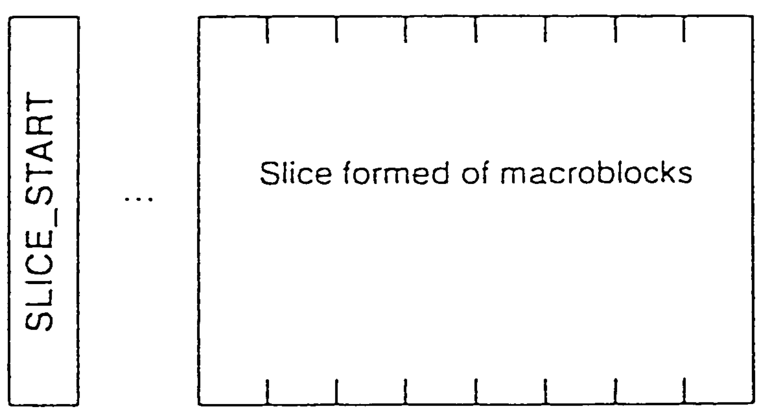


FIG. 104

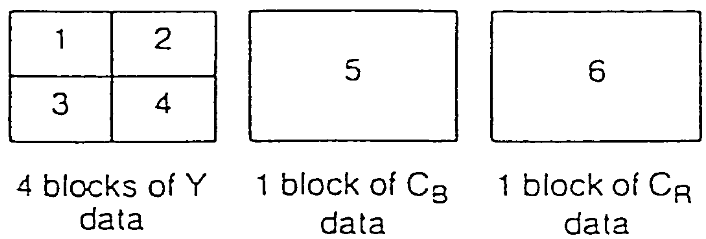


FIG. 105

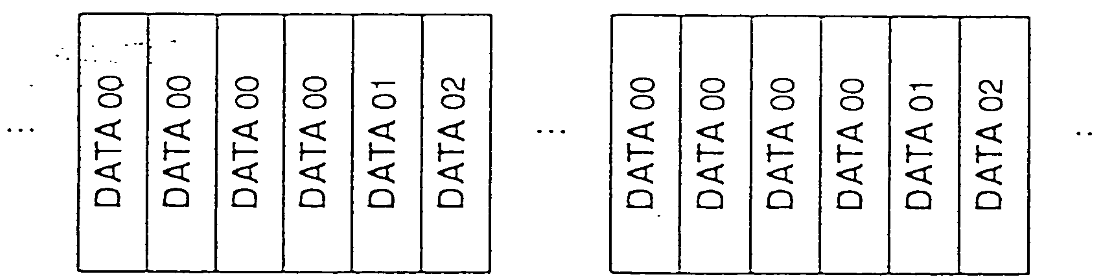


FIG. 106

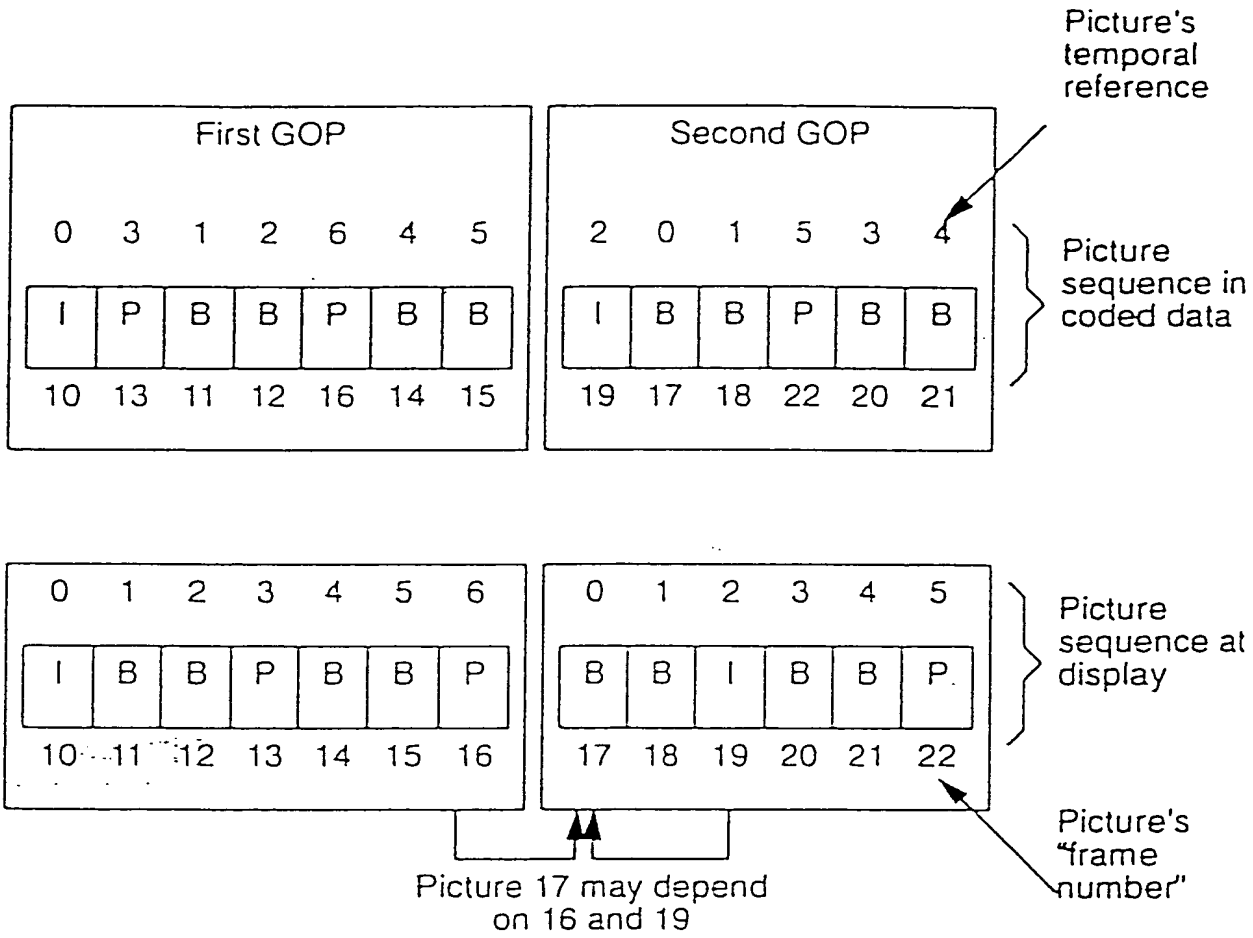

Picture's temporal reference
First GOP
Second GOP
0 3 1 2 6 4 5
I P B B P B B
10 13 11 12 16 14 15
2 0 1 5 3 4
I B B P B B
19 17 18 22 20 21
Picture sequence in coded data
0 1 2 3 4 5 6
I B B P B B P
10 11 12 13 14 15 16
0 1 2 3 4 5
B B I B B P
17 18 19 20 21 22
Picture sequence at display
Picture's "frame number"
Picture 17 may depend on 16 and 19

FIG. 107

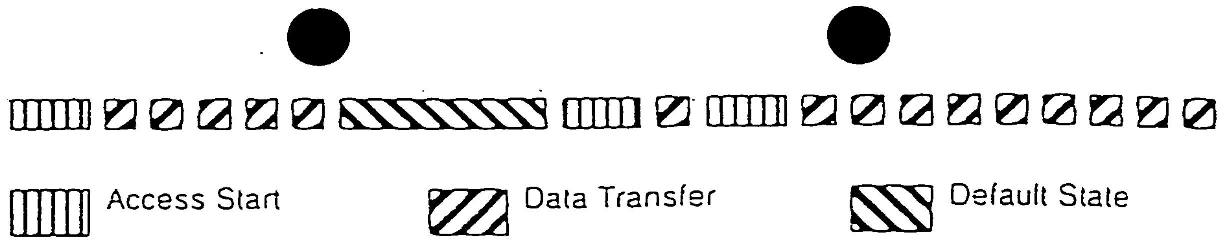


FIG. 108

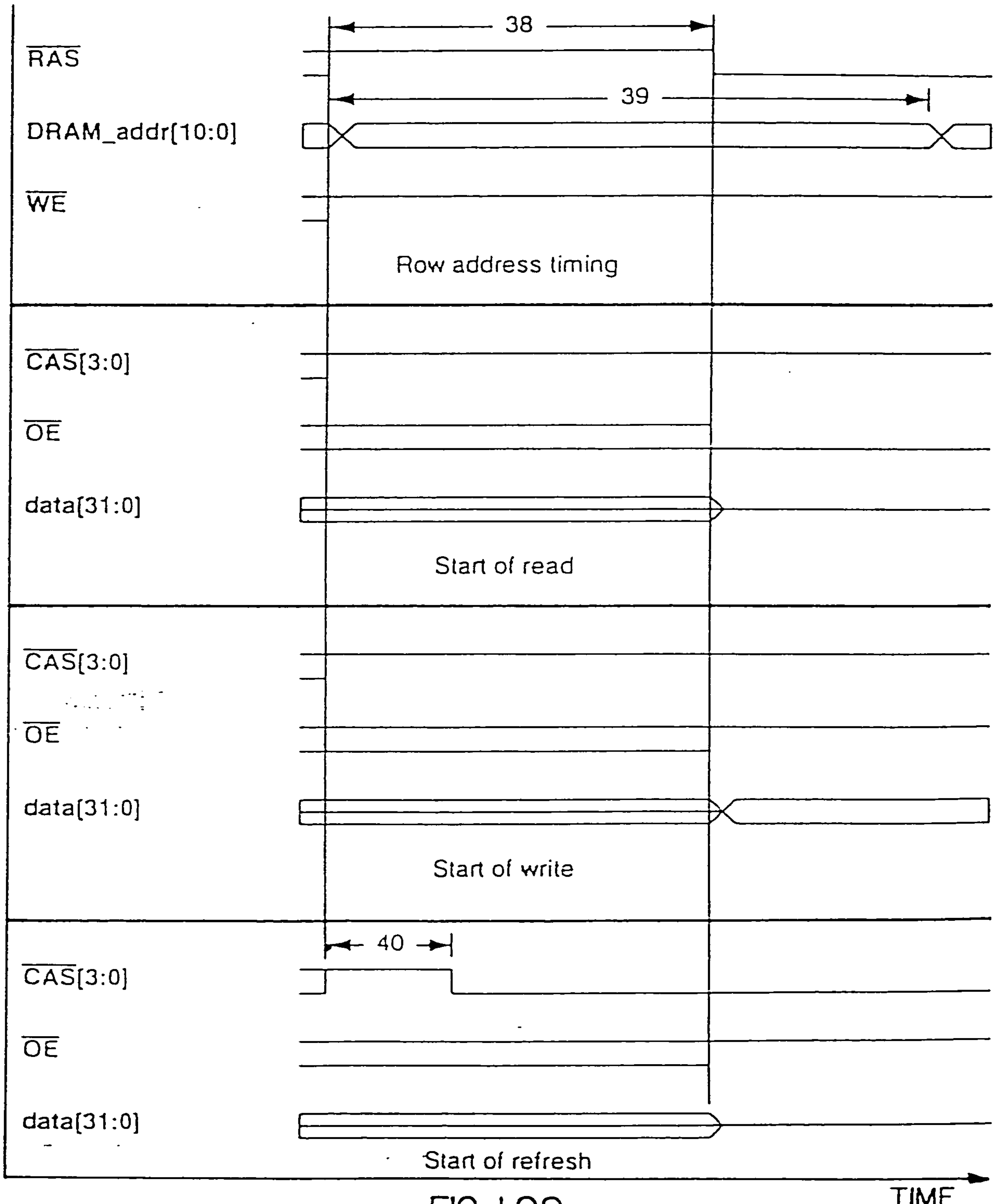


FIG. 109

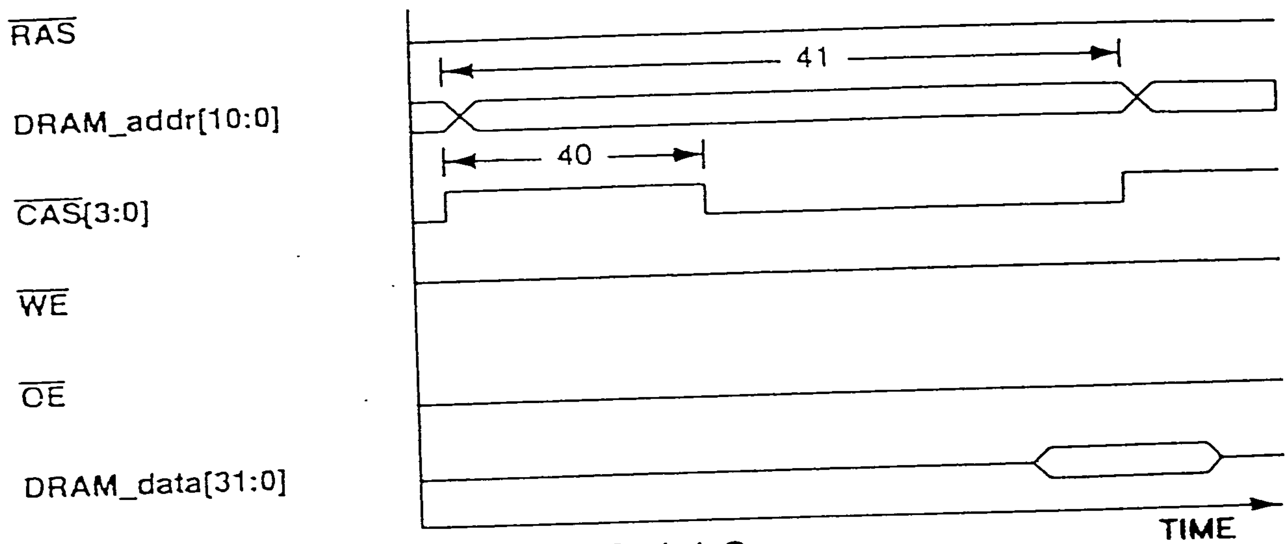


FIG. 110

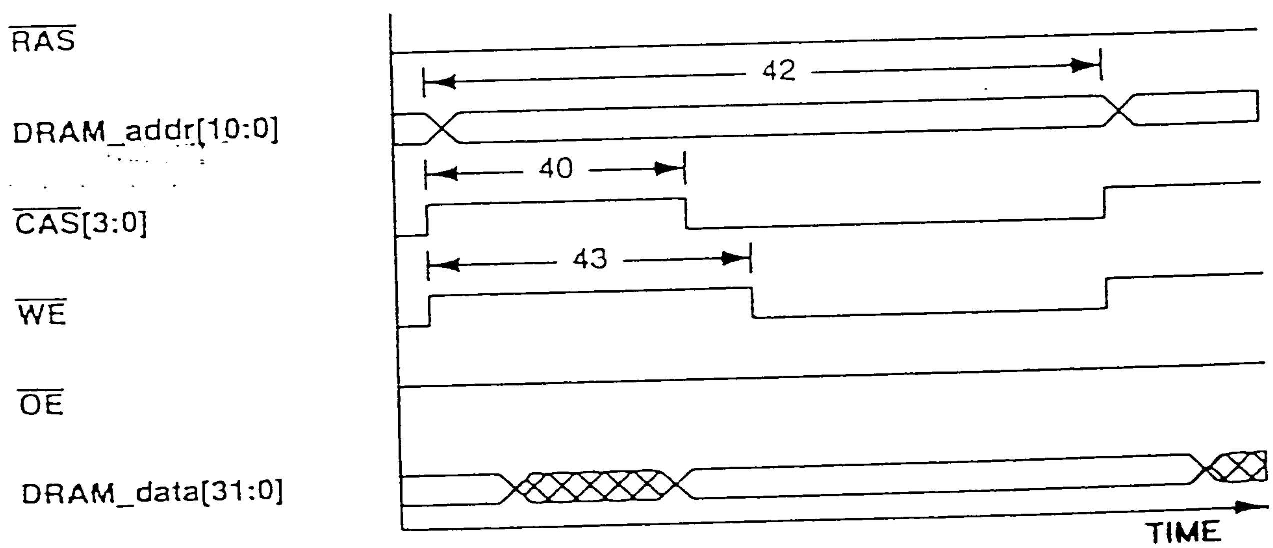


FIG. 111

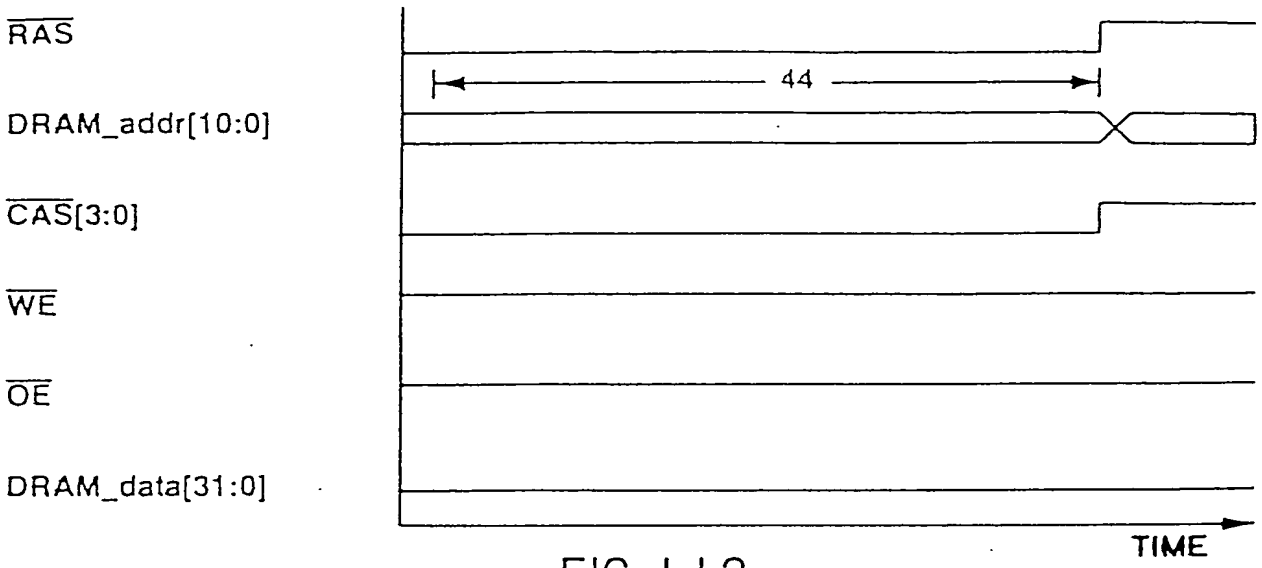


FIG. 112

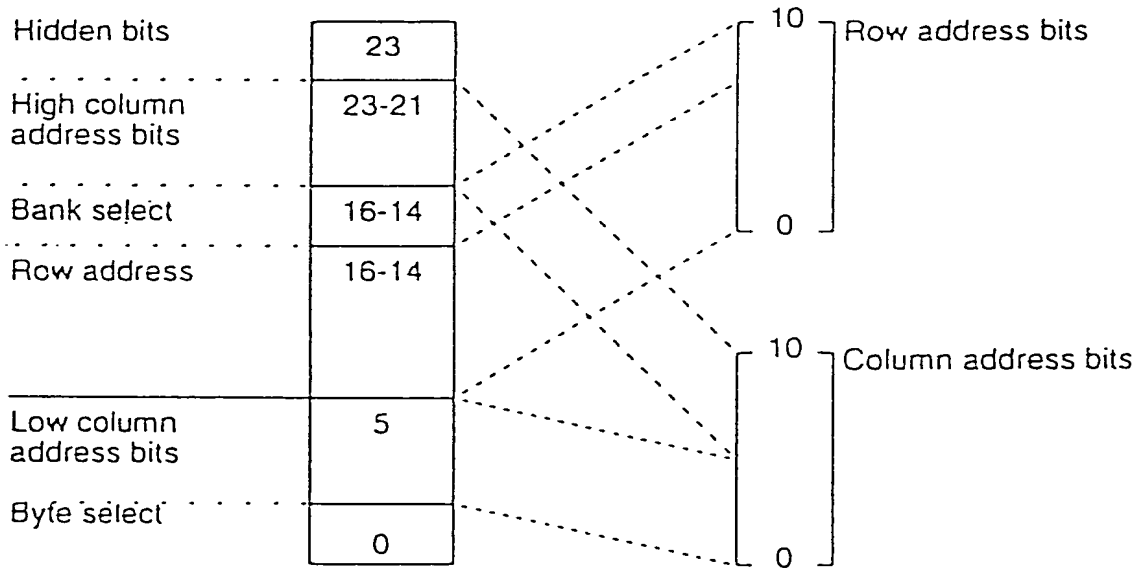


FIG. 113

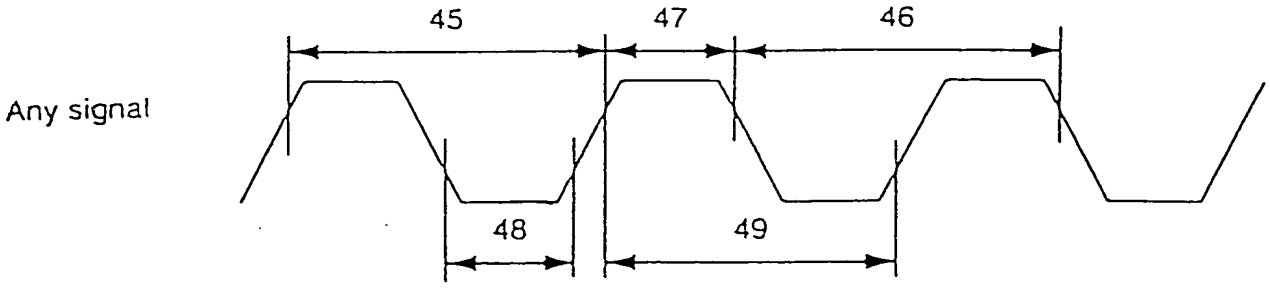


FIG. 114

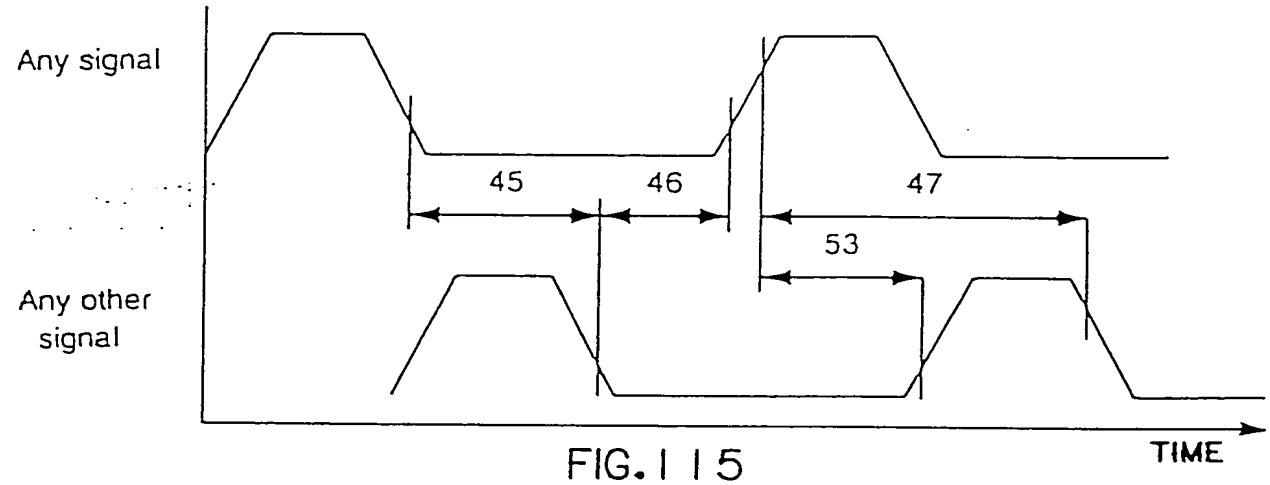


FIG. 115

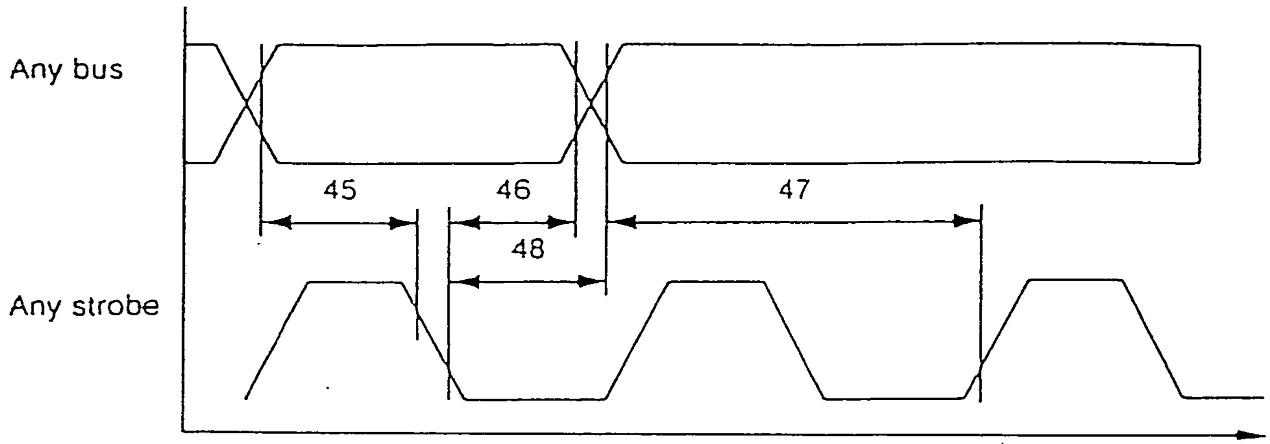


FIG. 116

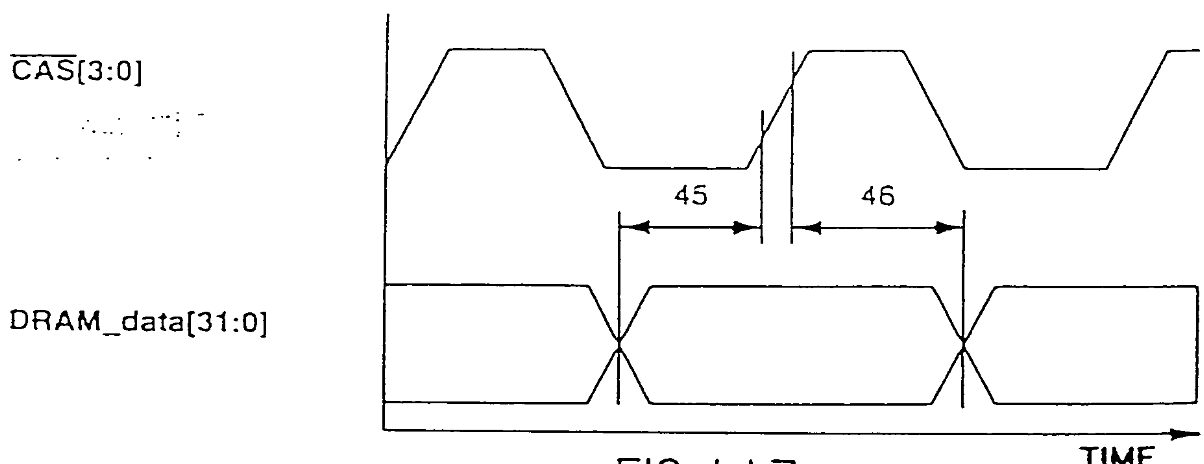


FIG. 117

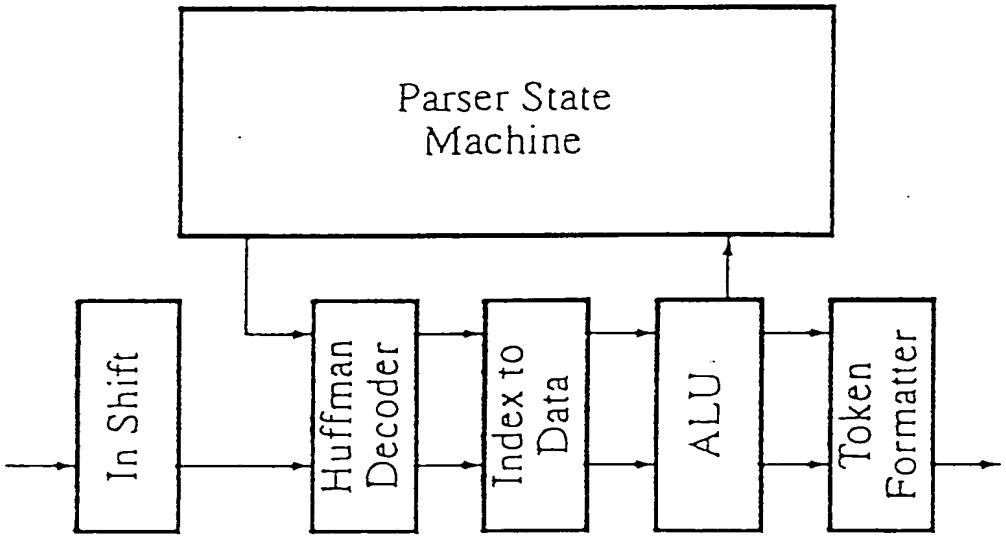


FIG. 118

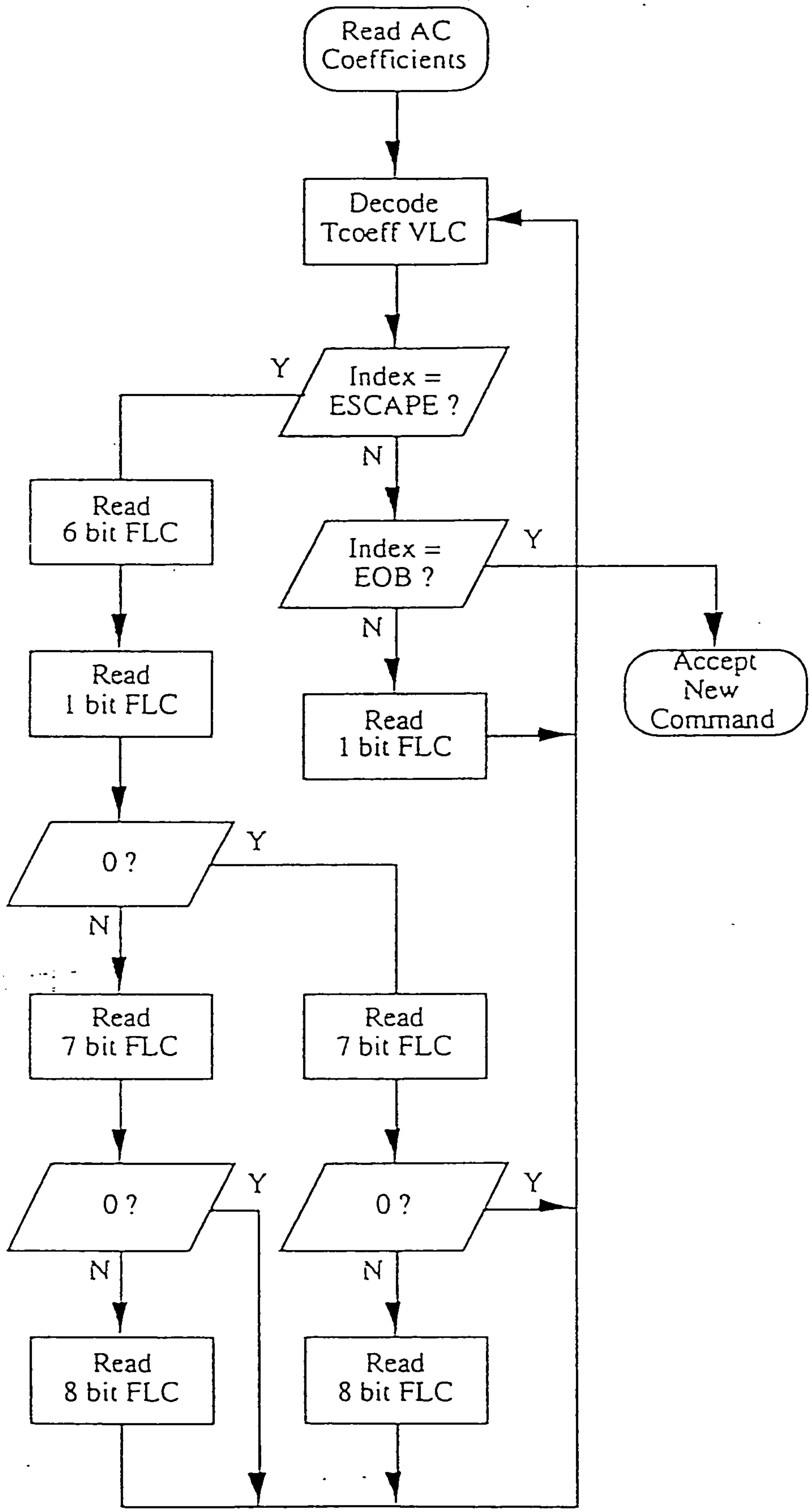

Read AC Coefficients
Decode Tcoeff VLC
Index = ESCAPE ?
Y
N
Index = EOB ?
Y
N
Accept New Command
Read 1 bit FLC
Read 6 bit FLC
Read 1 bit FLC
0 ?
Y
N
Read 7 bit FLC
Read 7 bit FLC
0 ?
Y
N
0 ?
Y
N
Read 8 bit FLC
Read 8 bit FLC

FIG. 119

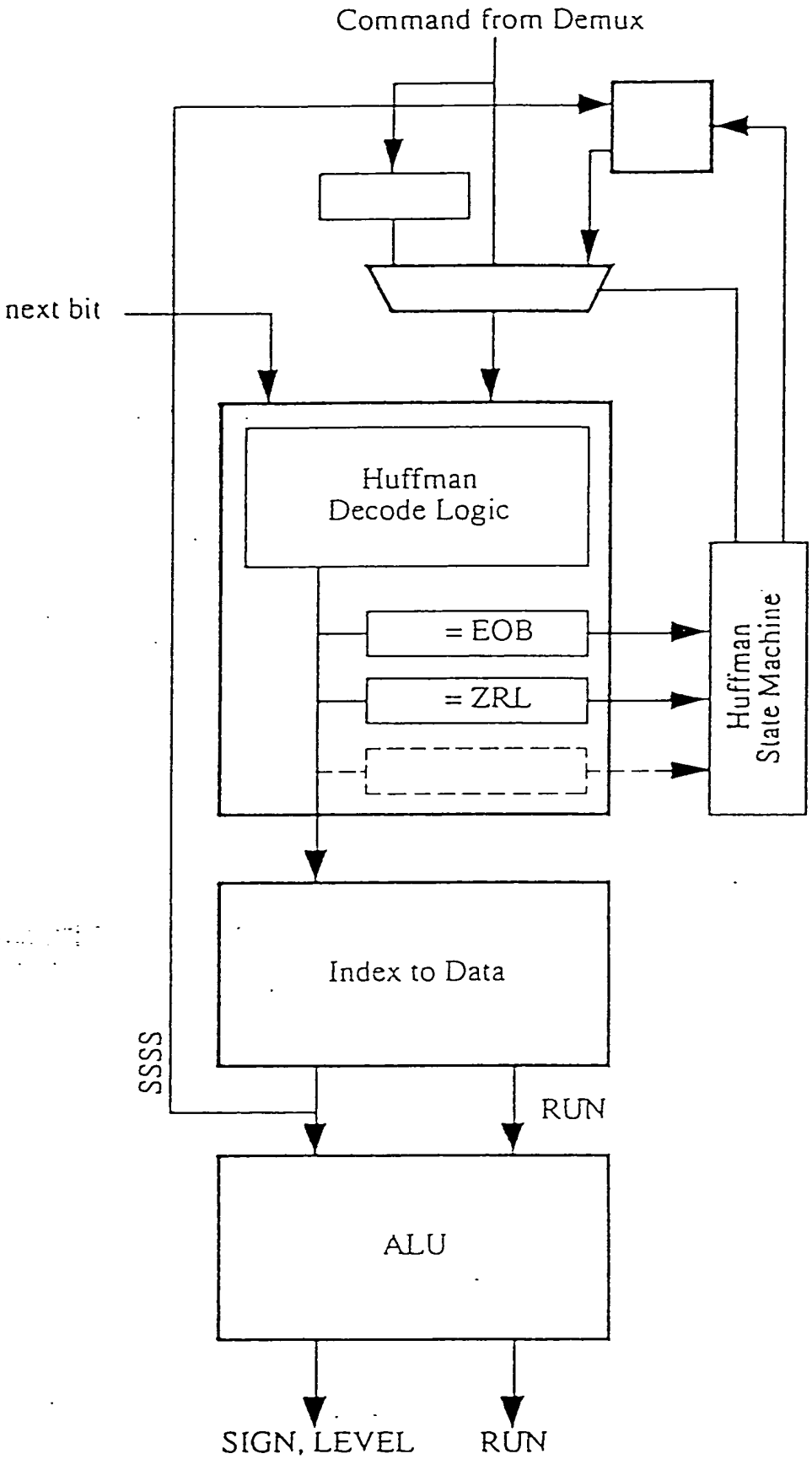


FIG. 120

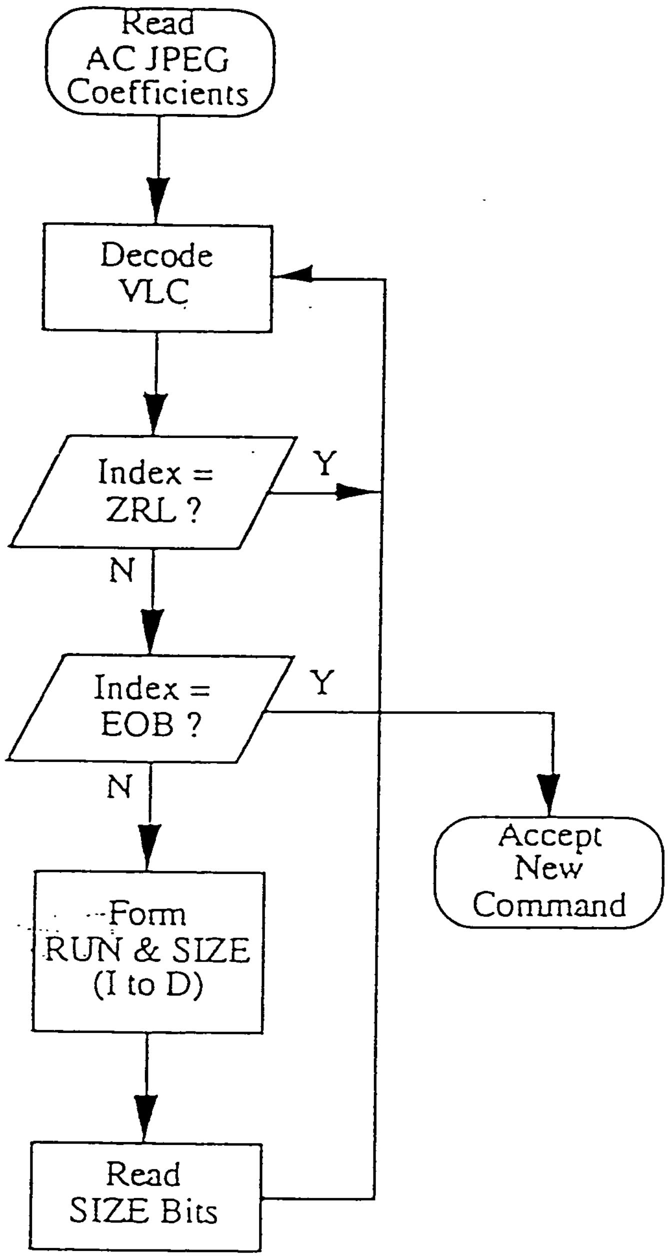


FIG. 121A

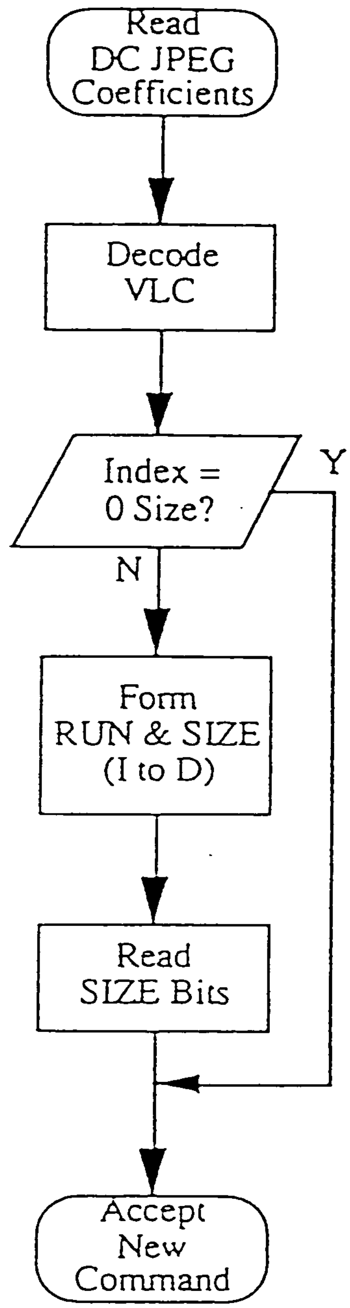


FIG. 121B

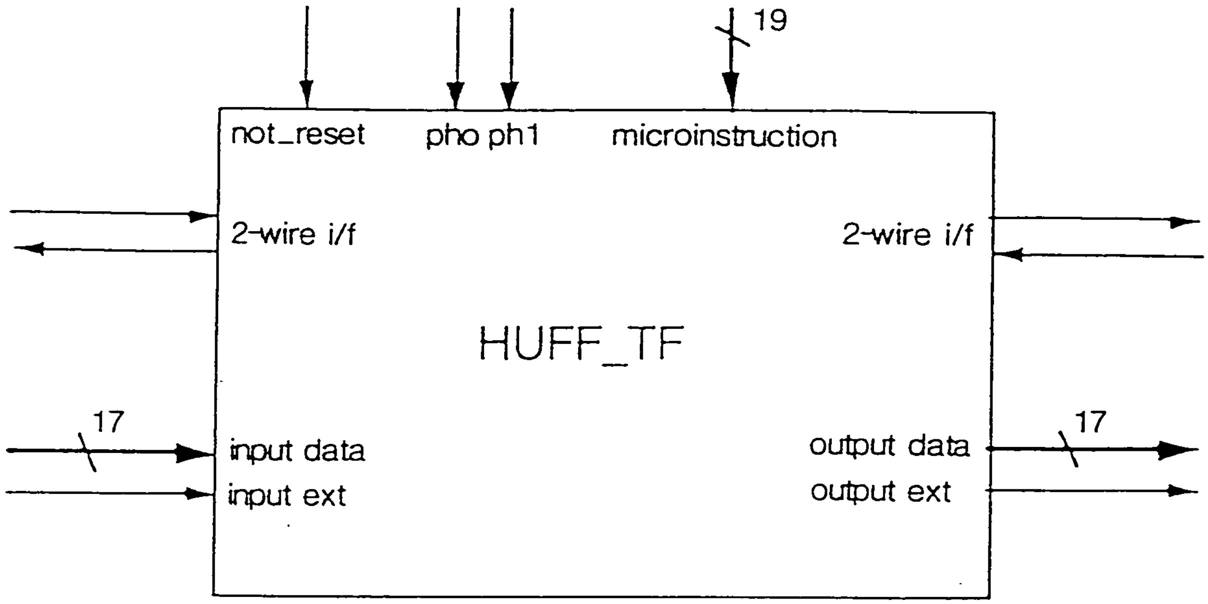


FIG. 122

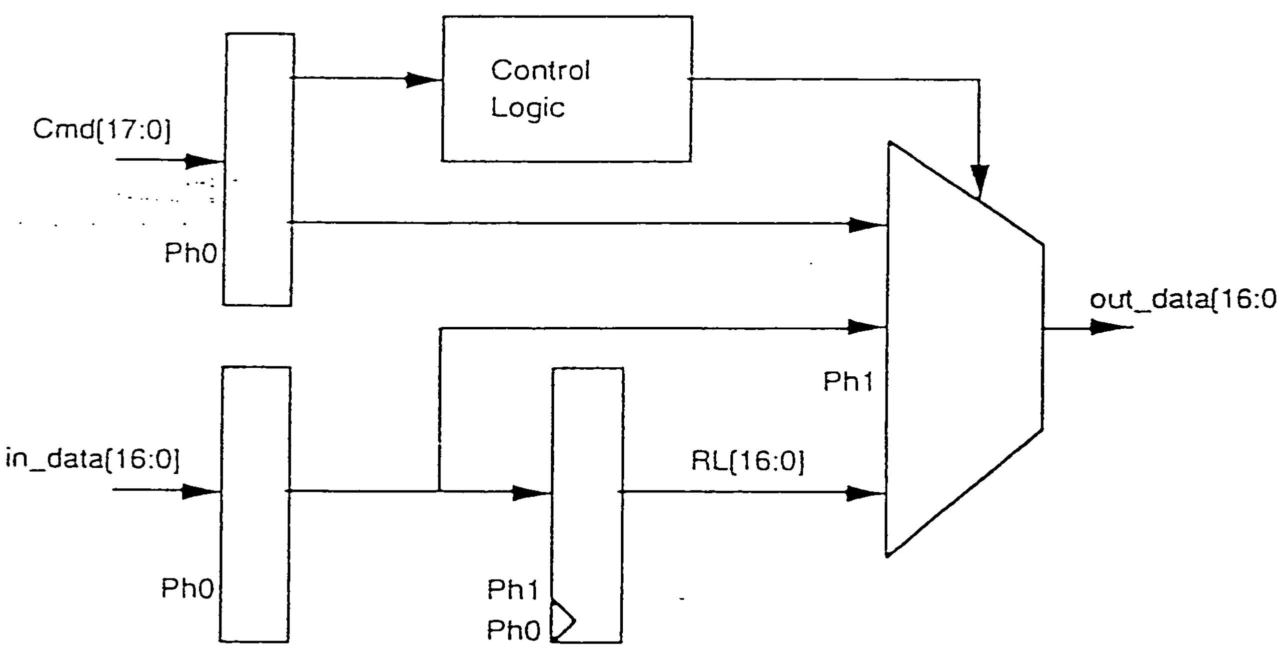


FIG. 123

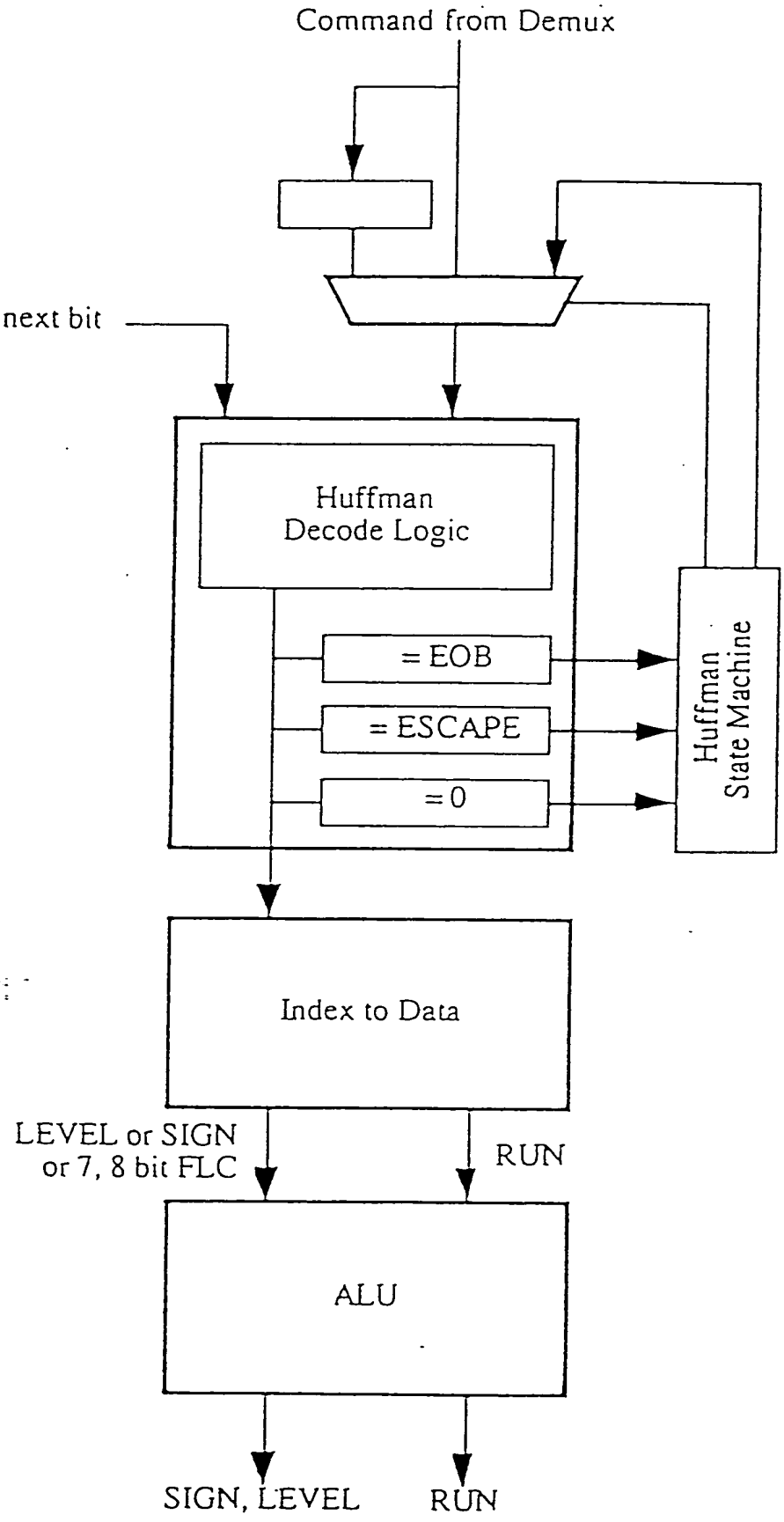

Command from Demux
next bit
Huffman
Decode Logic
= EOB
= ESCAPE
= 0
Huffman
State Machine
Index to Data
LEVEL or SIGN
or 7, 8 bit FLC
RUN
ALU
SIGN, LEVEL
RUN

FIG. 124

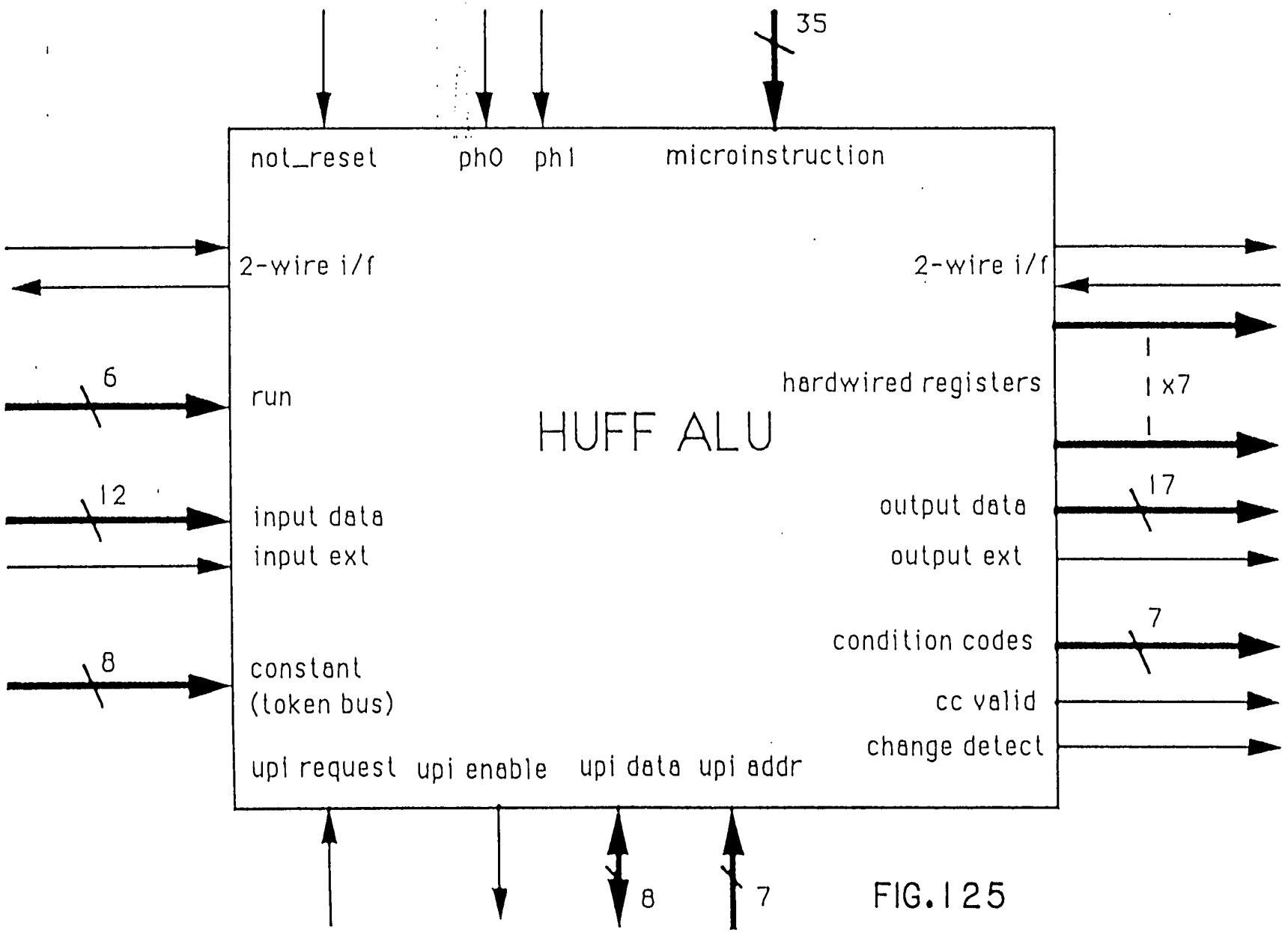

35
not_reset
ph0
ph1
microinstruction
2-wire i/f
2-wire i/f
1
6
run
hardwired registers
1 x7
HUFF ALU
1
12
input data
output data
17
input ext
output ext
7
condition codes
8
constant
(token bus)
cc valid
change detect
upi request
upi enable
upi data
upi addr
8
7

FIG.125

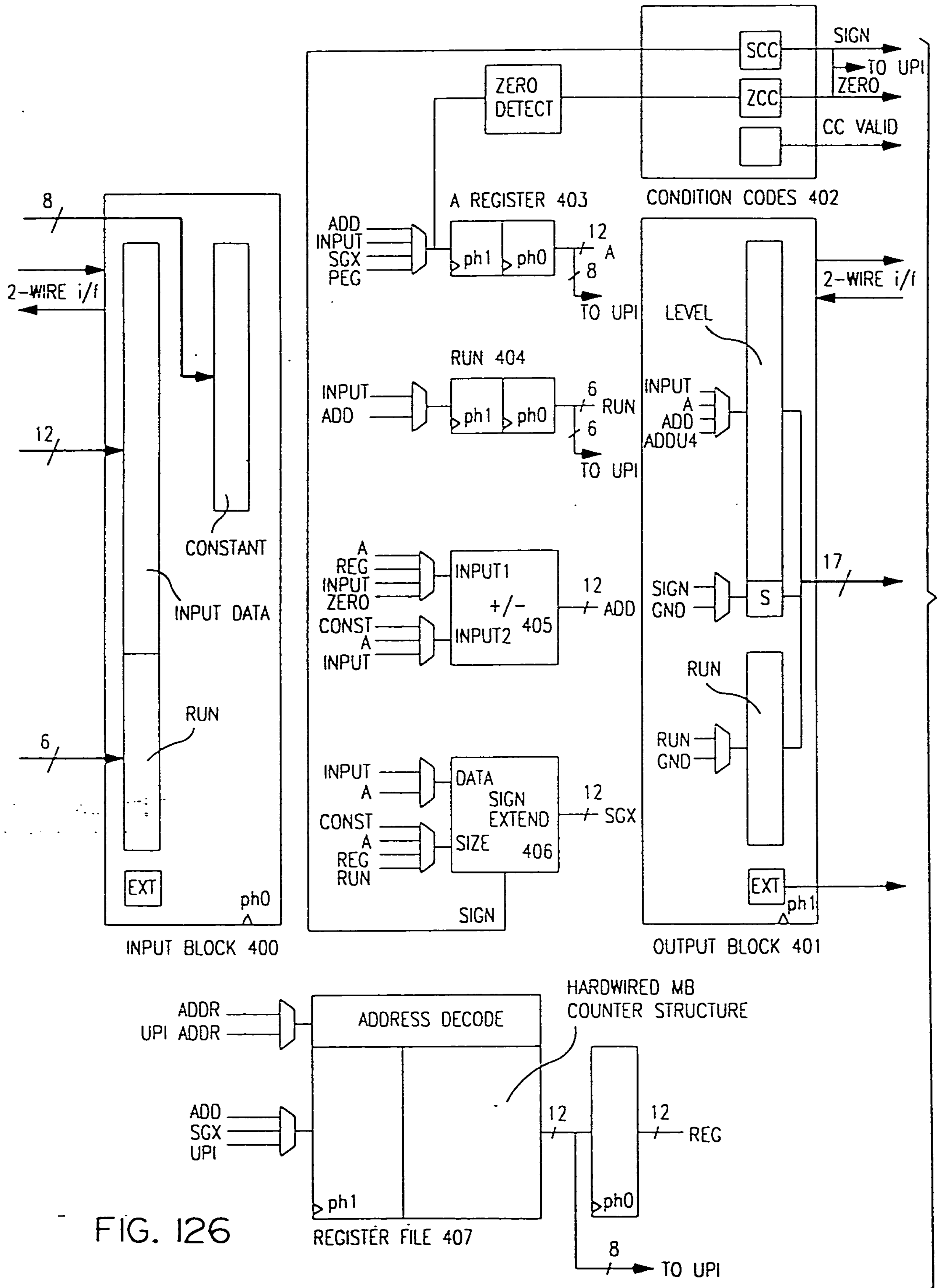

SIGN
TO UPI
ZERO
CC VALID
SCC
ZCC
ZERO DETECT
CONDITION CODES 402
A REGISTER 403
ADD
INPUT
SGX
PEG
ph1
ph0
12
A
8
TO UPI
2-WIRE i/f
LEVEL
RUN 404
INPUT
ADD
6
RUN
6
TO UPI
INPUT
A
ADD
ADDU4
CONSTANT
INPUT DATA
RUN
A
REG
INPUT
ZERO
CONST
A
INPUT
INPUT1
+/-
INPUT2
405
12
ADD
17
SIGN
GND
S
RUN
RUN
GND
INPUT
A
CONST
A
REG
RUN
DATA
SIGN EXTEND
SIZE
406
12
SGX
SIGN
EXT
ph0
INPUT BLOCK 400
EXT
ph1
OUTPUT BLOCK 401
HARDWIRED MB COUNTER STRUCTURE
ADDR
UPI ADDR
ADDRESS DECODE
ADD
SGX
UPI
ph1
REGISTER FILE 407
12
12
REG
ph0
8
TO UPI

FIG. 126

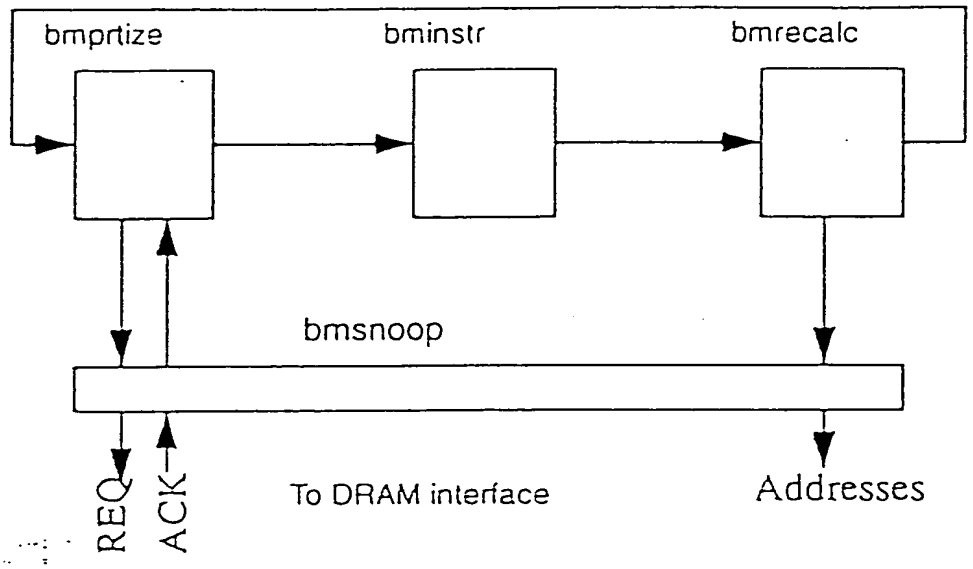


FIG. 127

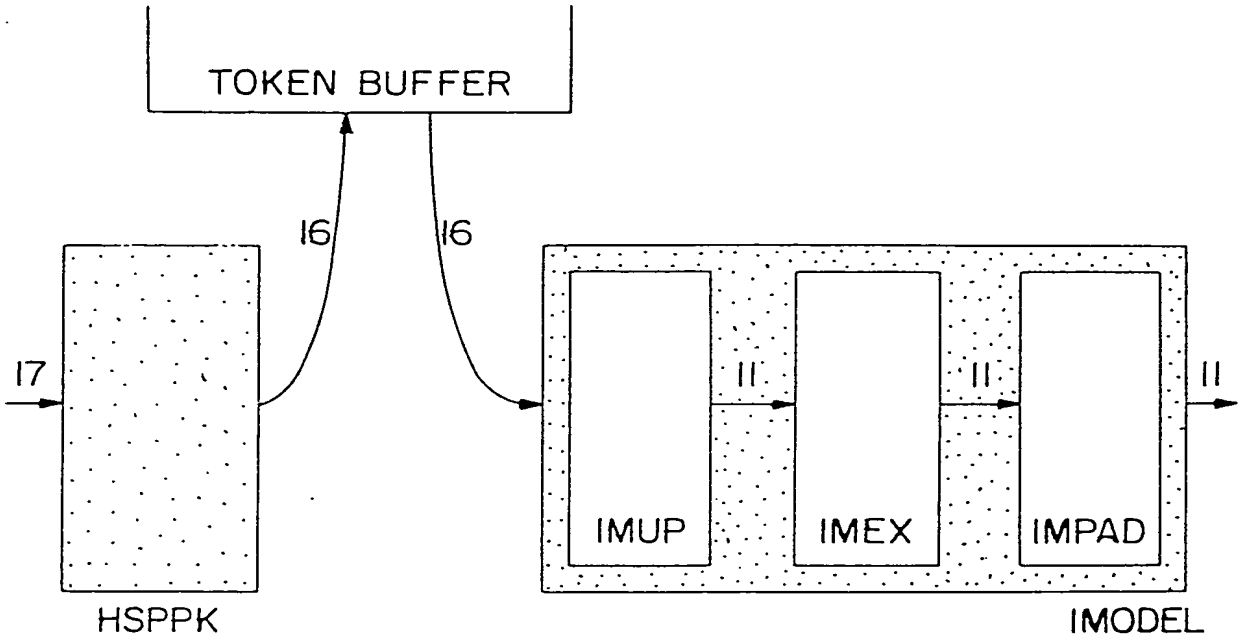


FIG. 128

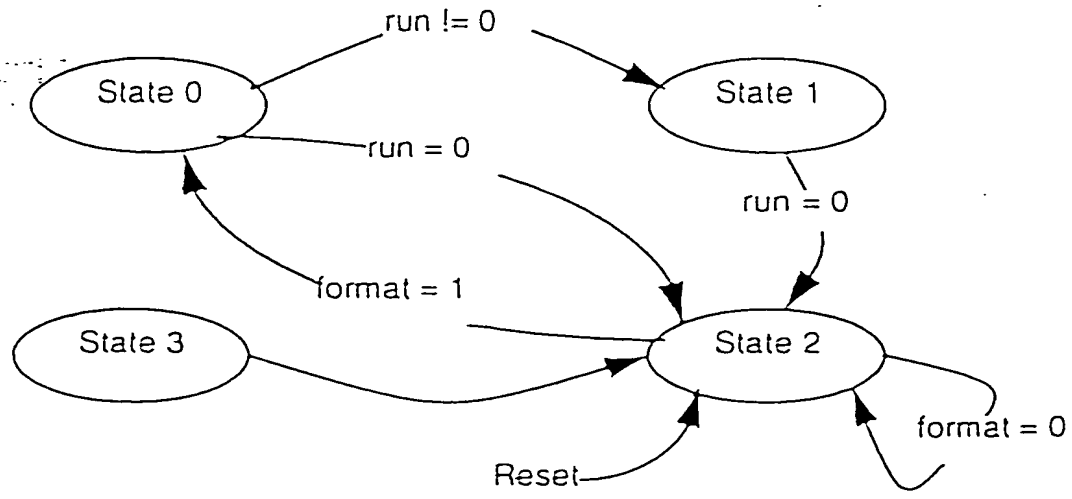


FIG. 129

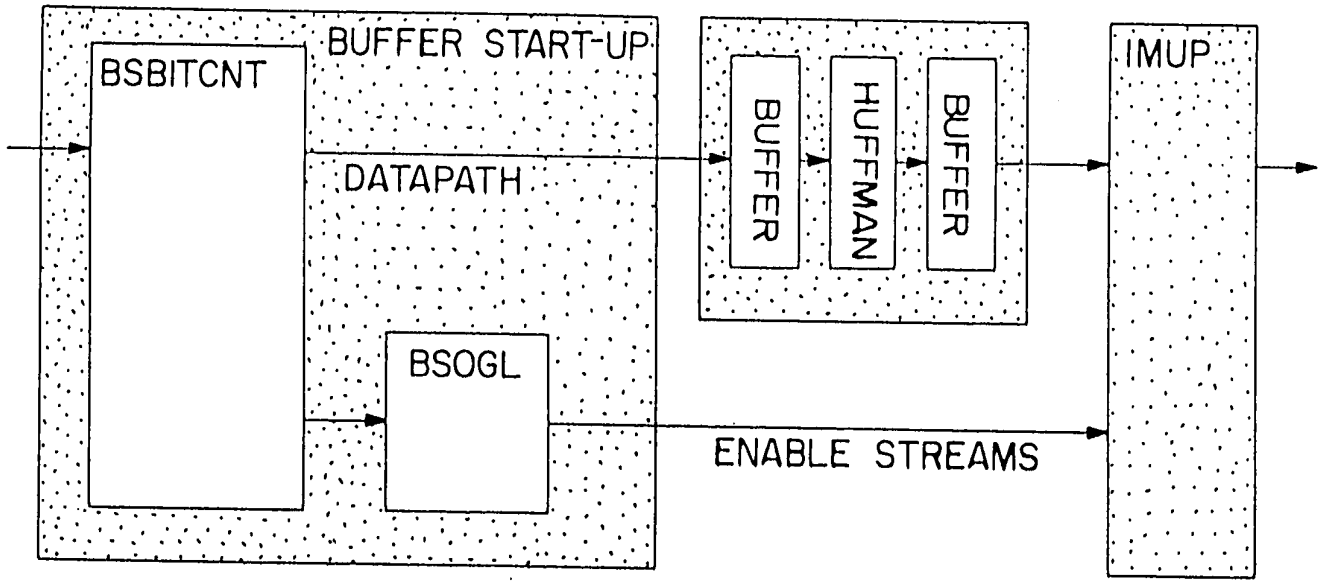

BUFFER START-UP
BSBITCNT
DATAPATH
BSOGL
BUFFER
HUFFMAN
BUFFER
IMUP
ENABLE STREAMS

FIG. 130

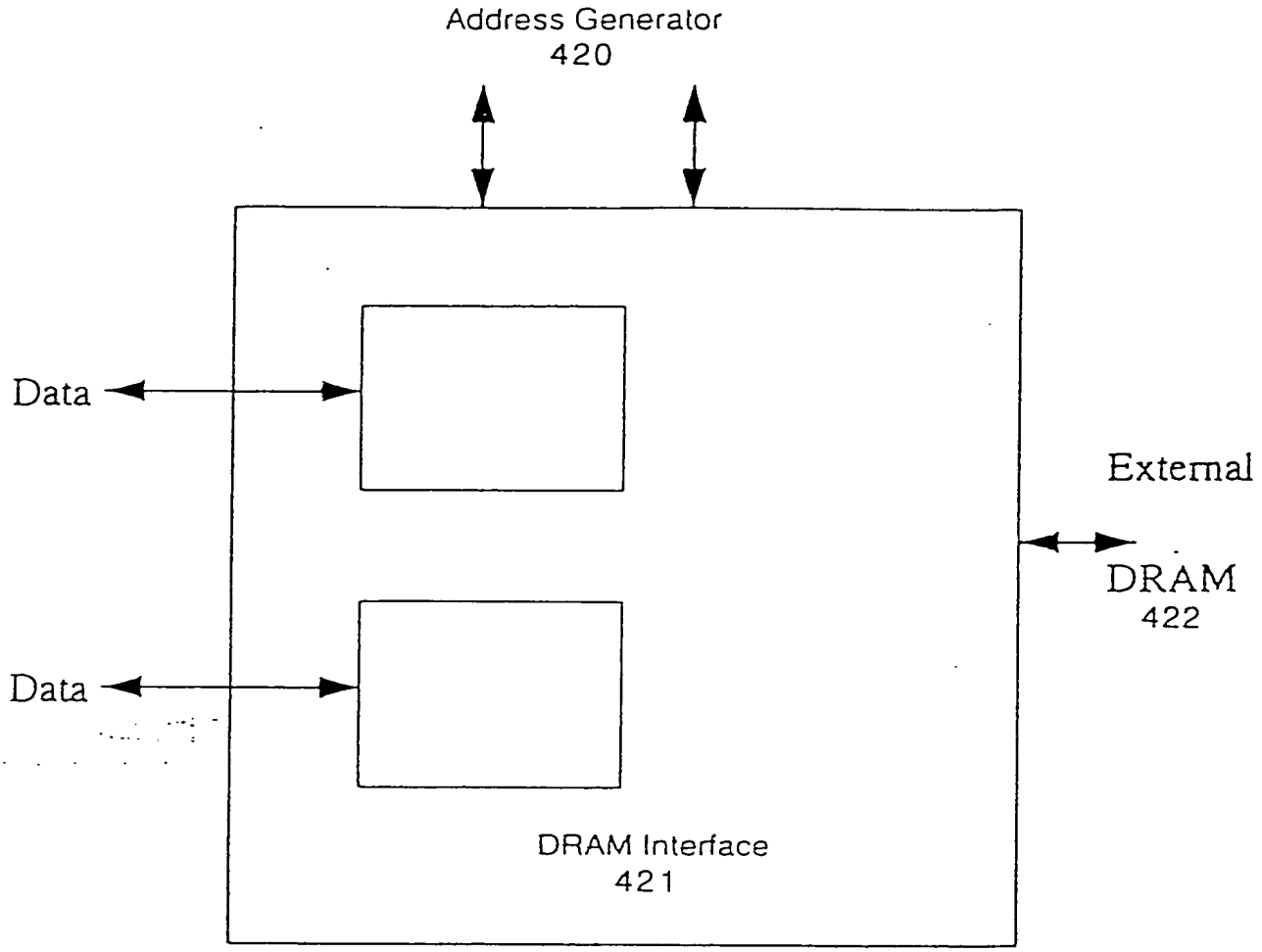

Address Generator
420
Data
Data
External
DRAM
422
DRAM Interface
421

FIG. 131

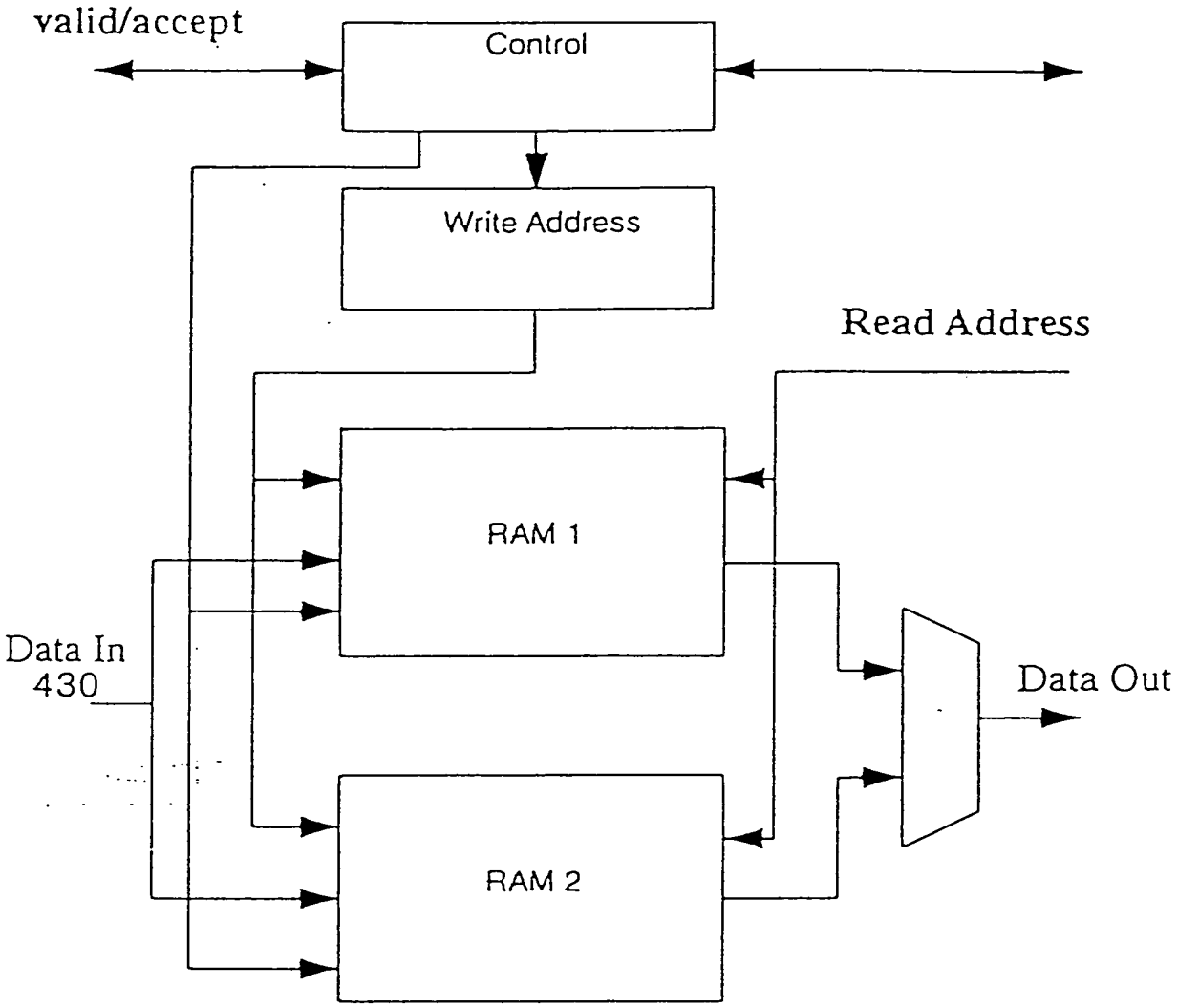

valid/accept
Control
Write Address
Read Address
RAM 1
Data In
430
Data Out
RAM 2

FIG. 132

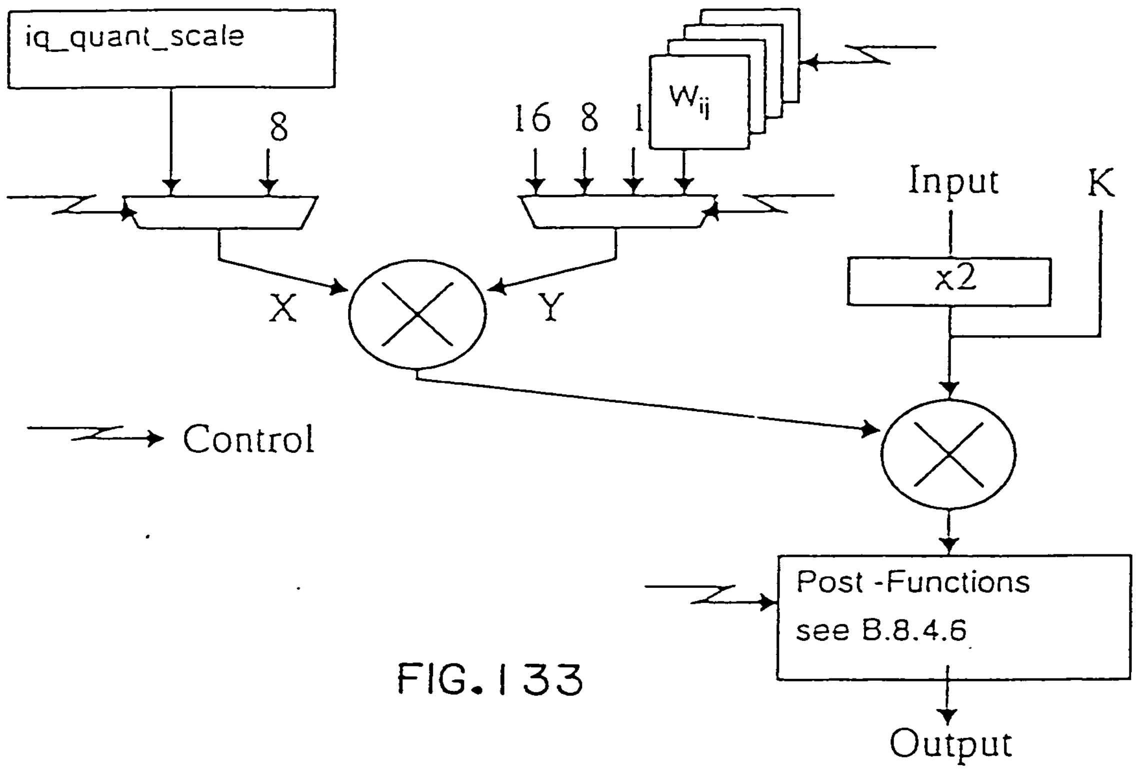


FIG. 133

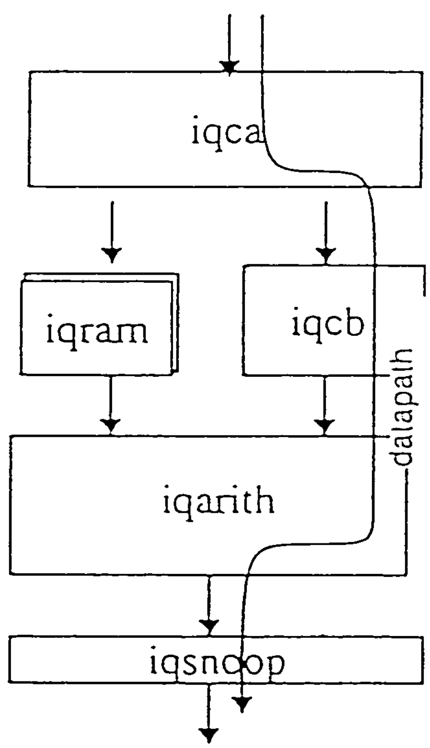


FIG. 134

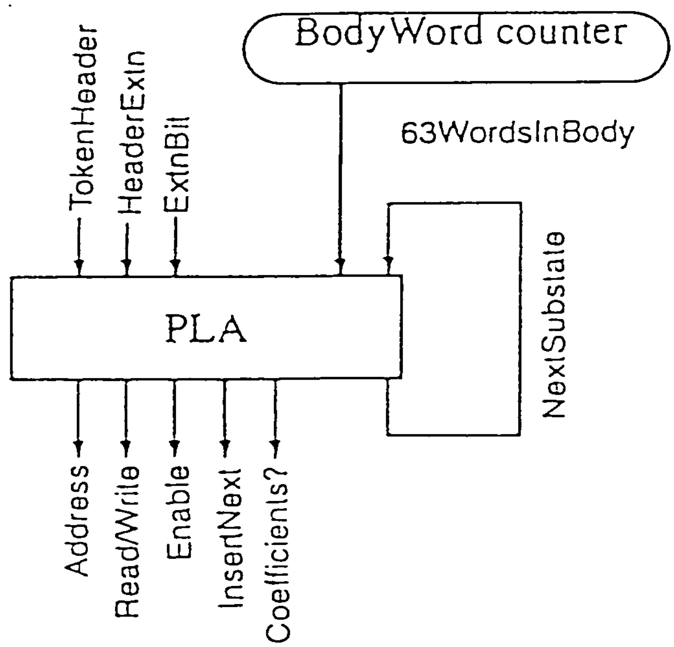

BodyWord counter
63WordsInBody
TokenHeader
HeaderExtn
ExtnBit
PLA
NextSubstate
Address
Read/Write
Enable
InsertNext
Coefficients?

FIG. 135

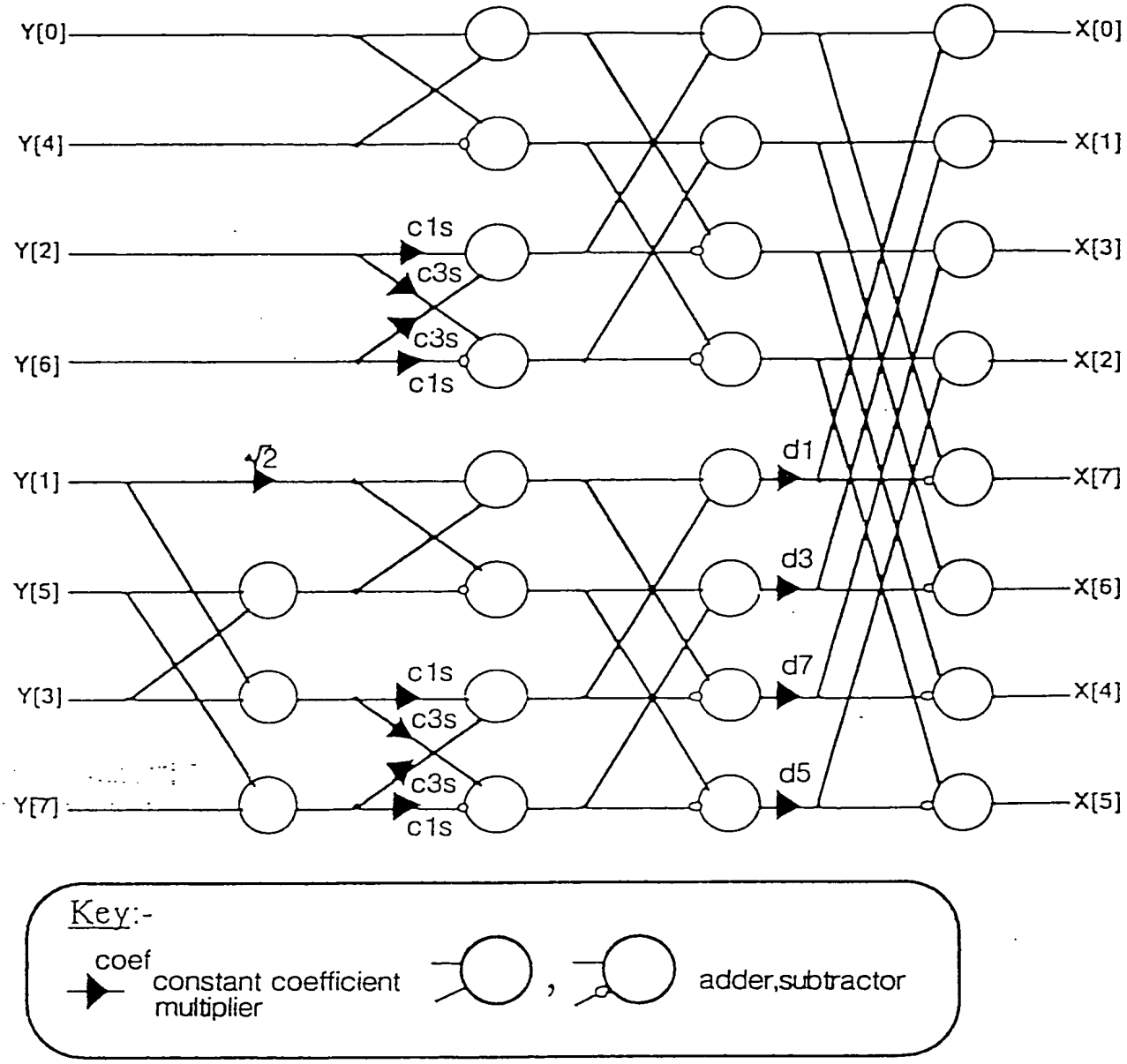

Y[0]
Y[4]
Y[2]
Y[6]
Y[1]
Y[5]
Y[3]
Y[7]
c1s
c3s
c3s
c1s
√2
c1s
c3s
c3s
c1s
d1
d3
d7
d5
X[0]
X[1]
X[3]
X[2]
X[7]
X[6]
X[4]
X[5]
Key:-
coef
constant coefficient multiplier
adder,subtractor

FIG.136

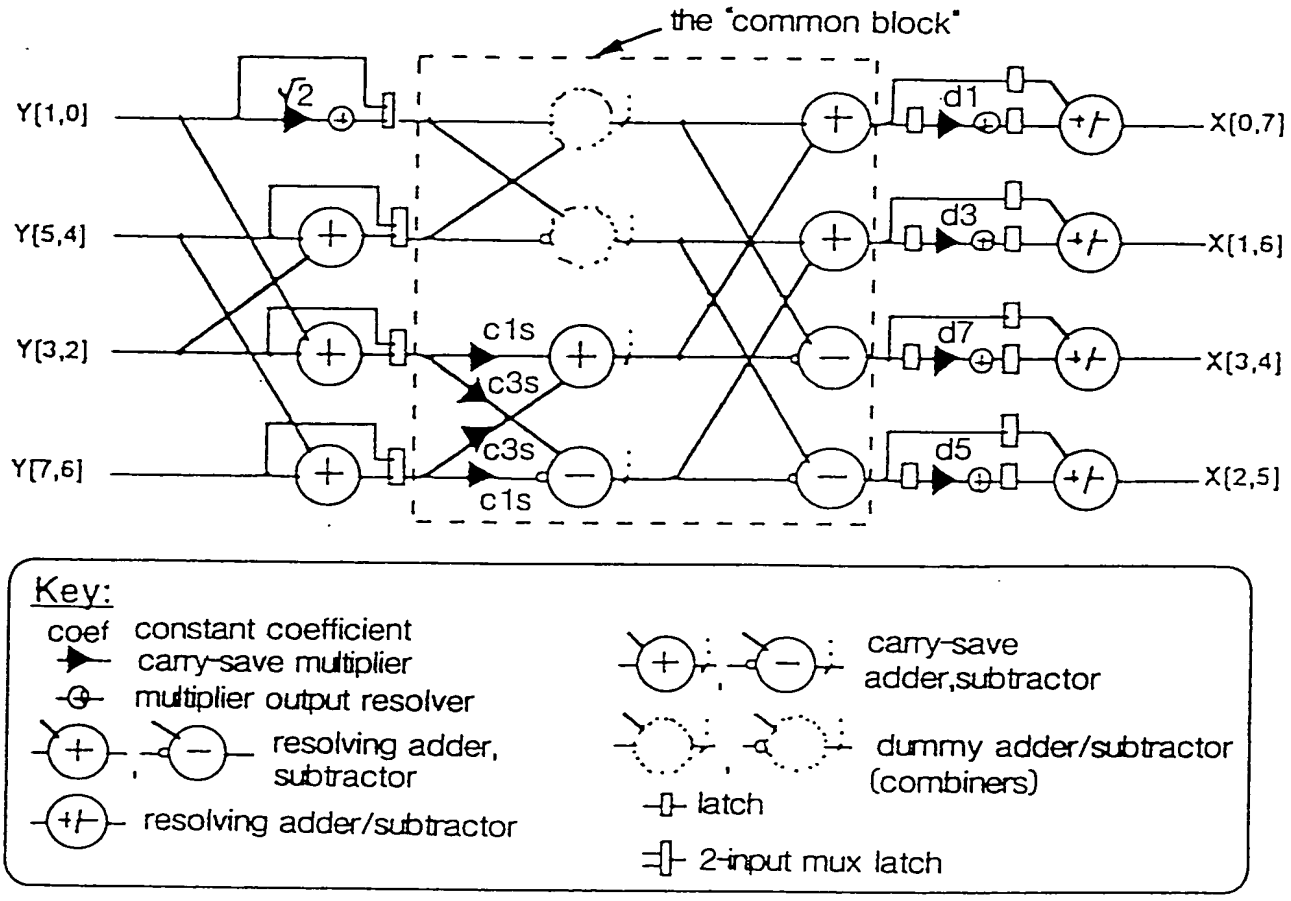


FIG. 137

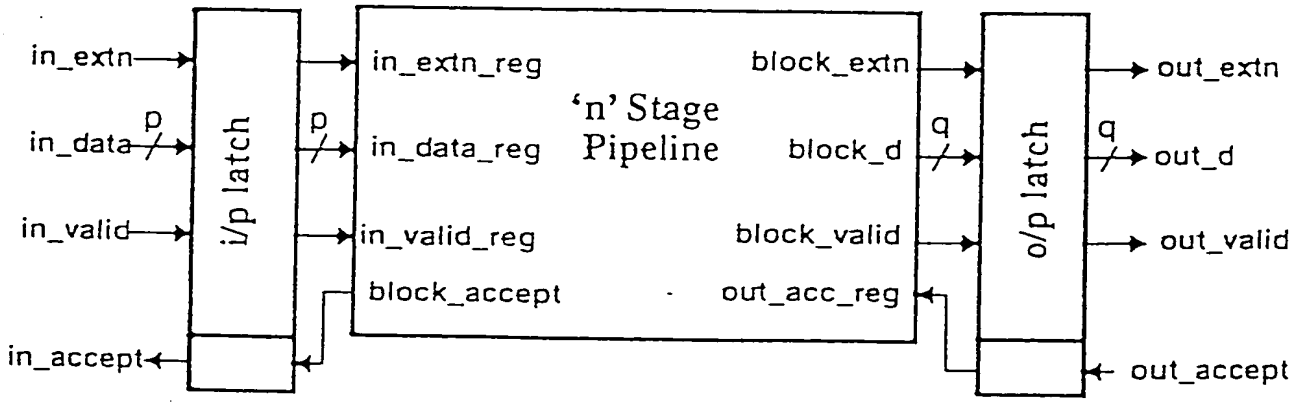


FIG. 138

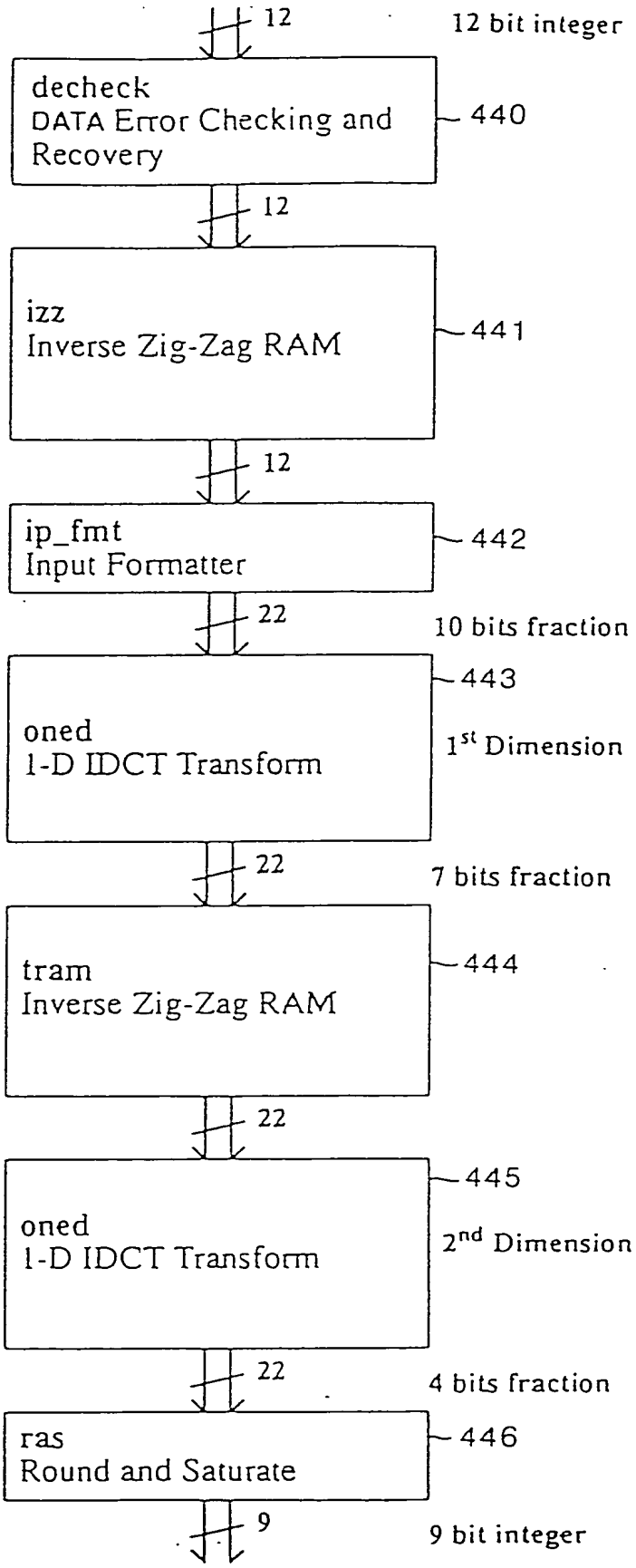


FIG. 139

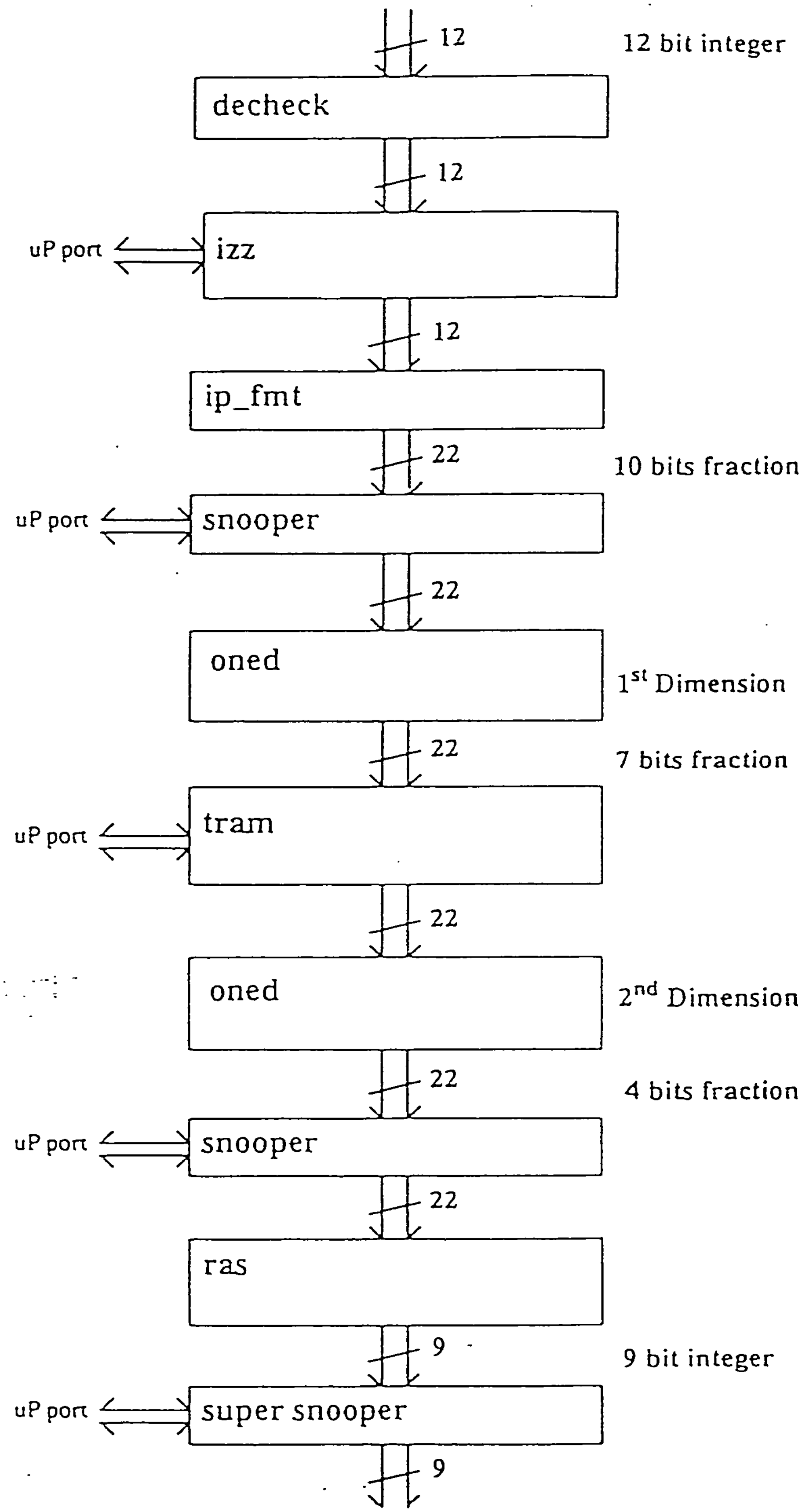
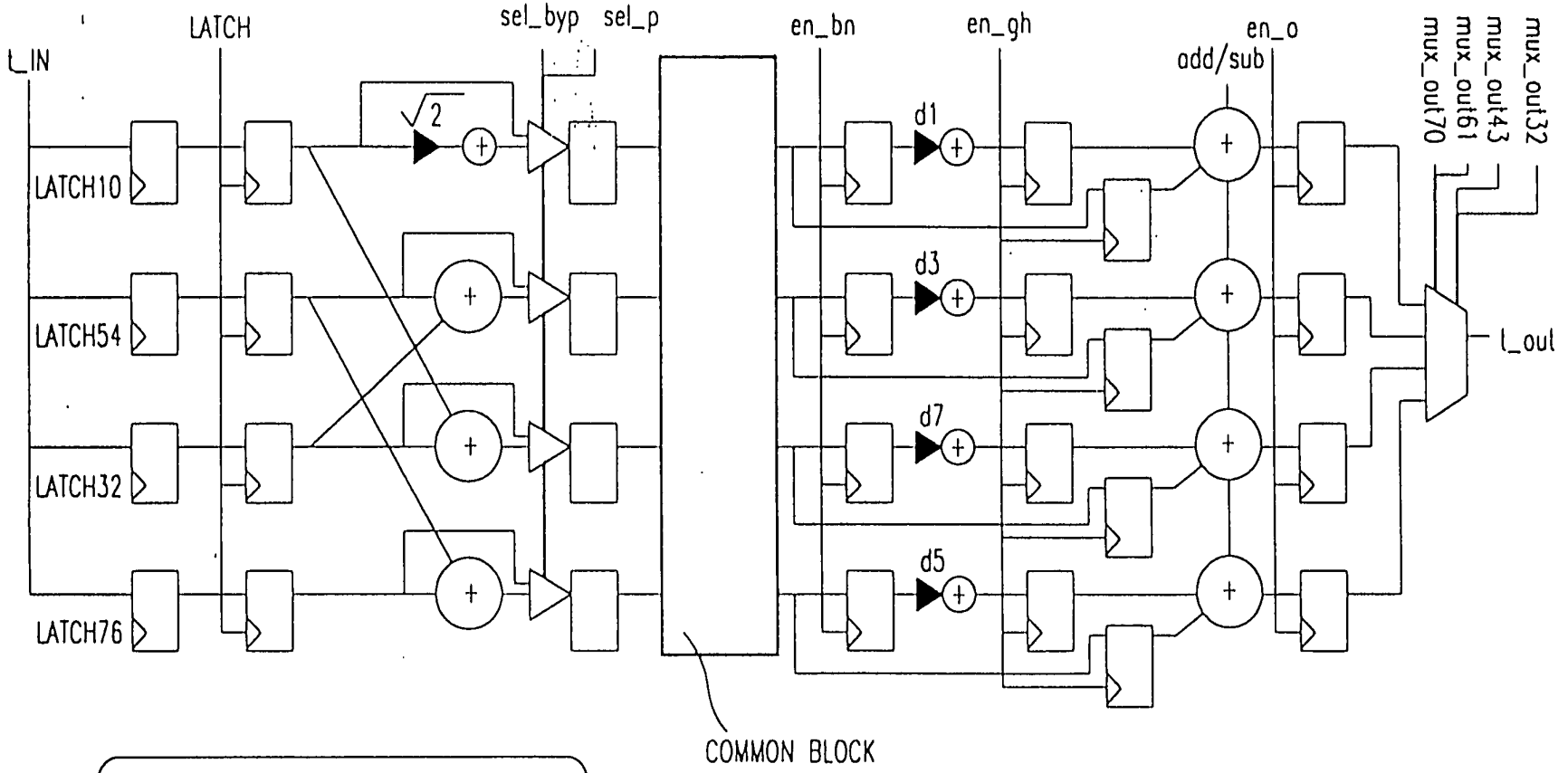
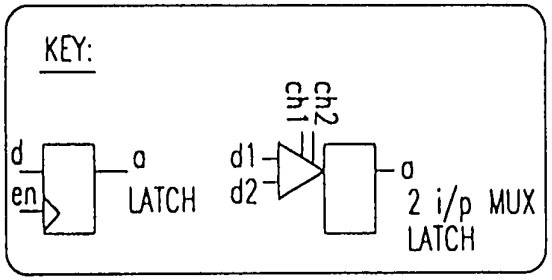


FIG. 140





NOTE: "COMMON BLOCK" IS ENTIRELY COMBINATIONAL (NO LATCHING)

FIG. 141

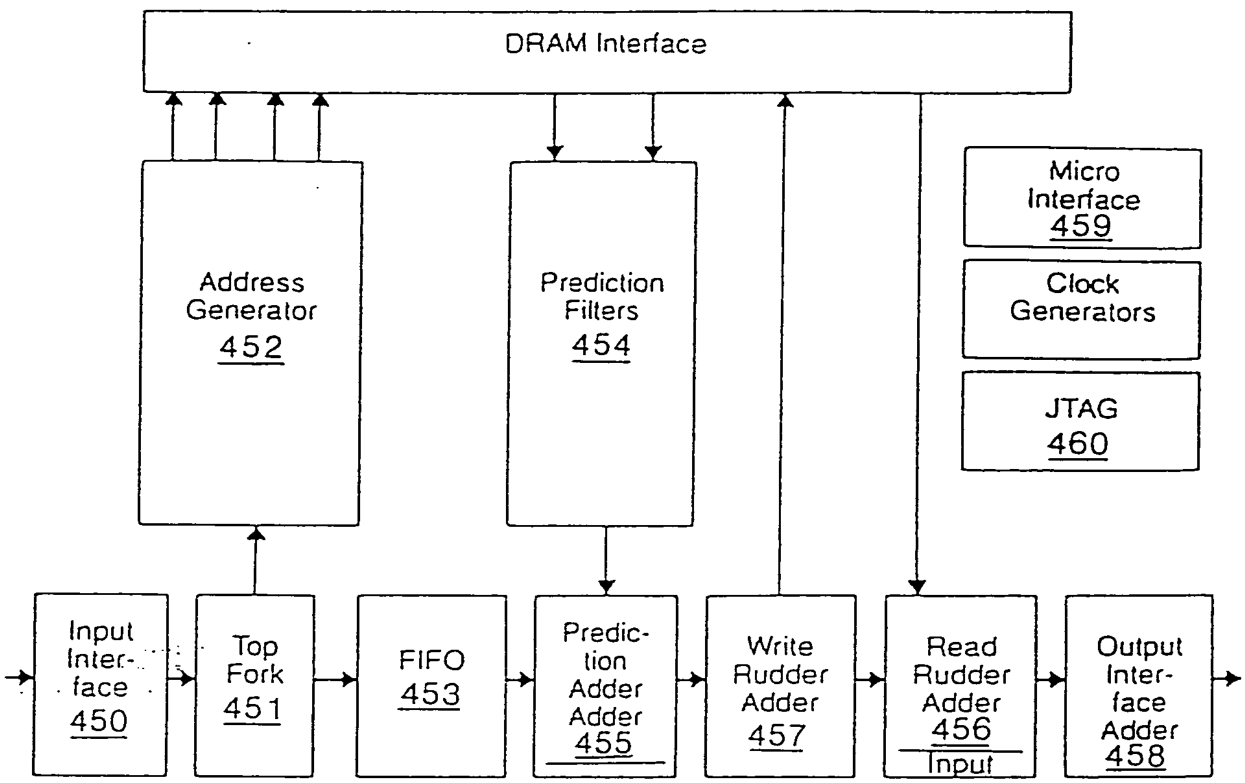


FIG. 142

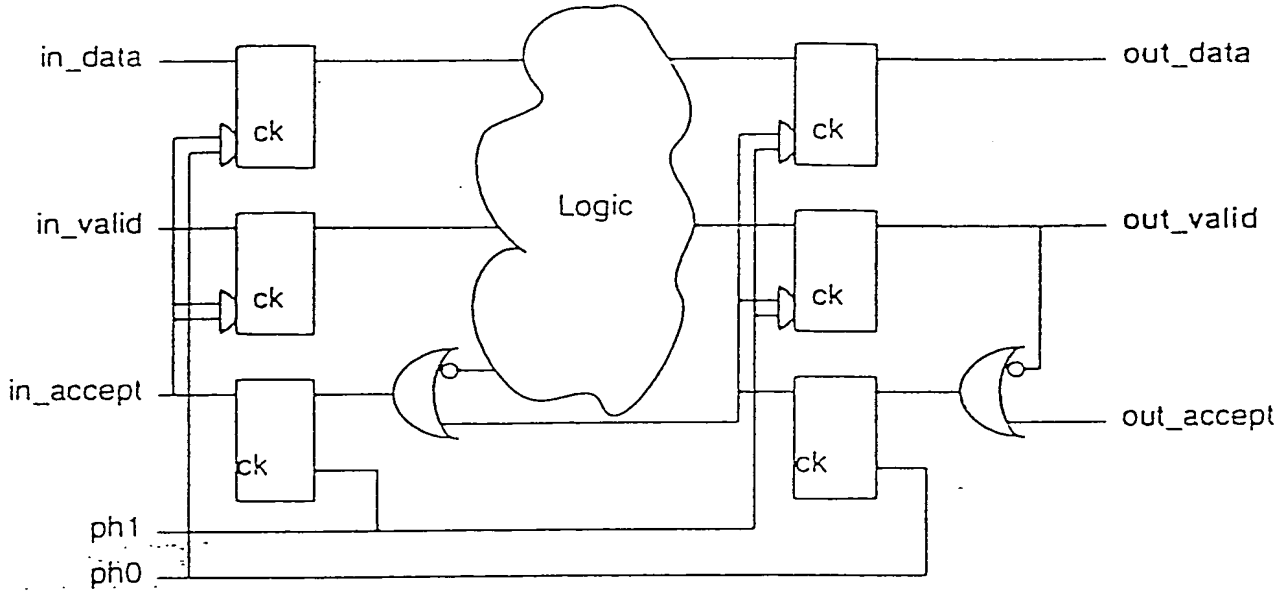


FIG. 143

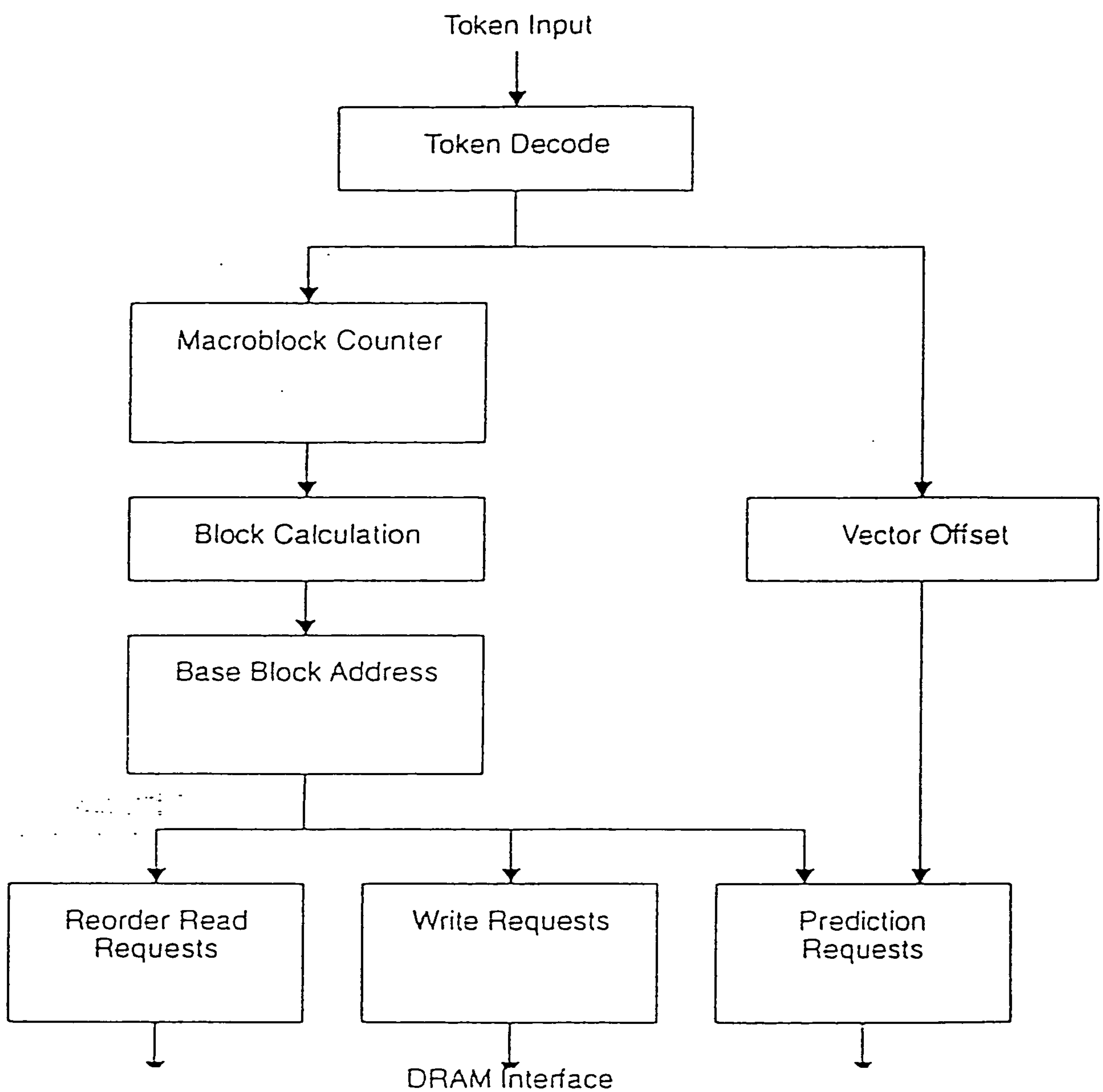

Token Input
Token Decode
Macroblock Counter
Block Calculation
Vector Offset
Base Block Address
Reorder Read Requests
Write Requests
Prediction Requests
DRAM Interface

FIG. 144

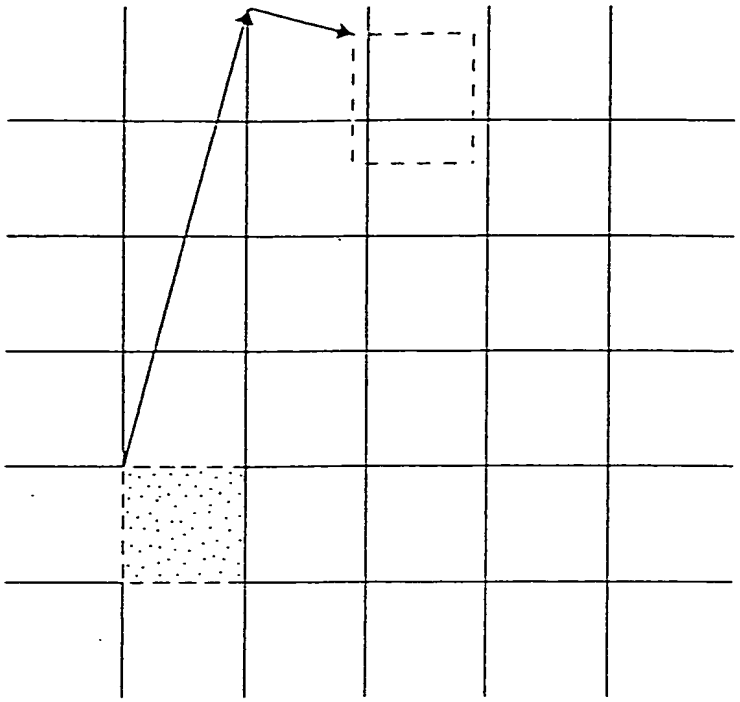


FIG. 145

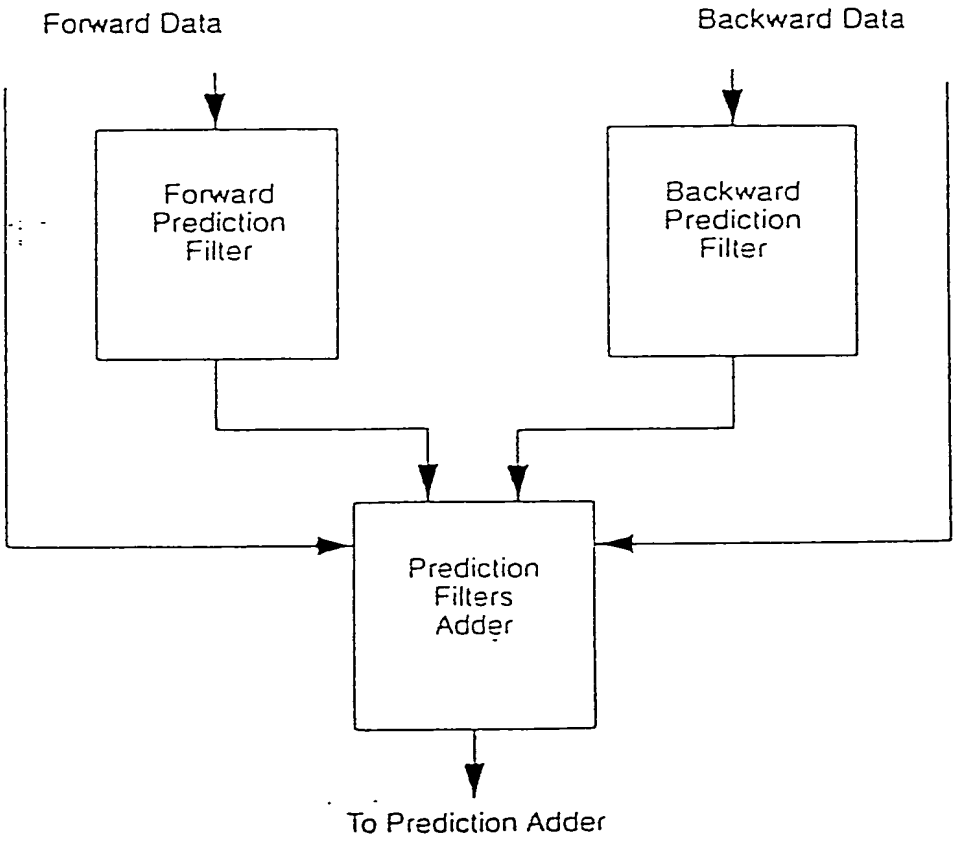


FIG. 146

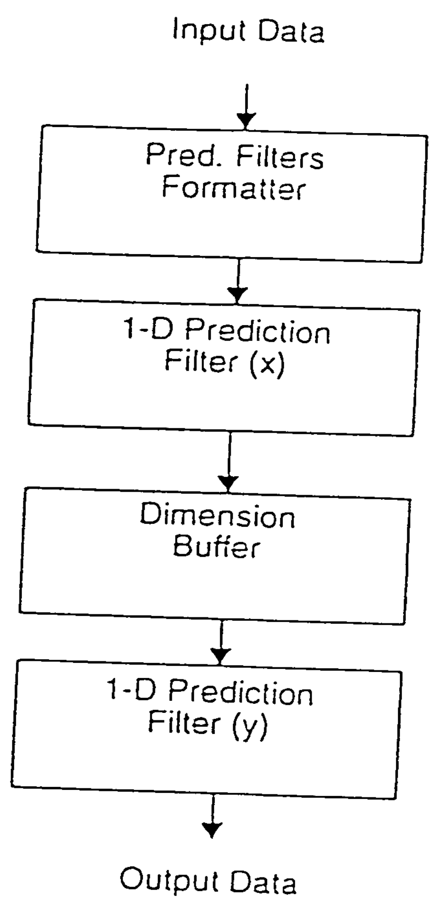

Input Data
Pred. Filters
Formatter
1-D Prediction
Filter (x)
Dimension
Buffer
1-D Prediction
Filter (y)
Output Data

FIG. 147

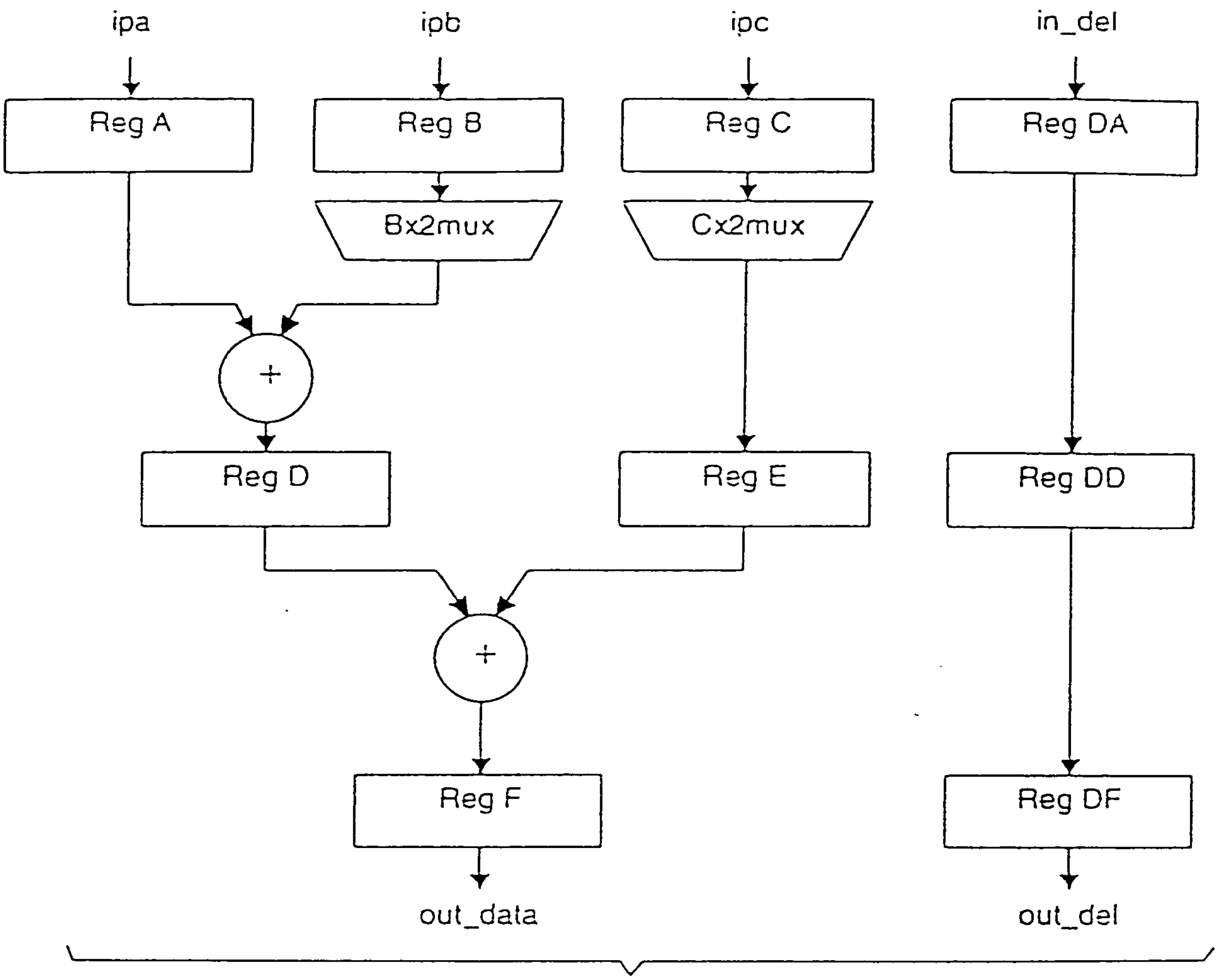


FIG. 148

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

FIG. 149

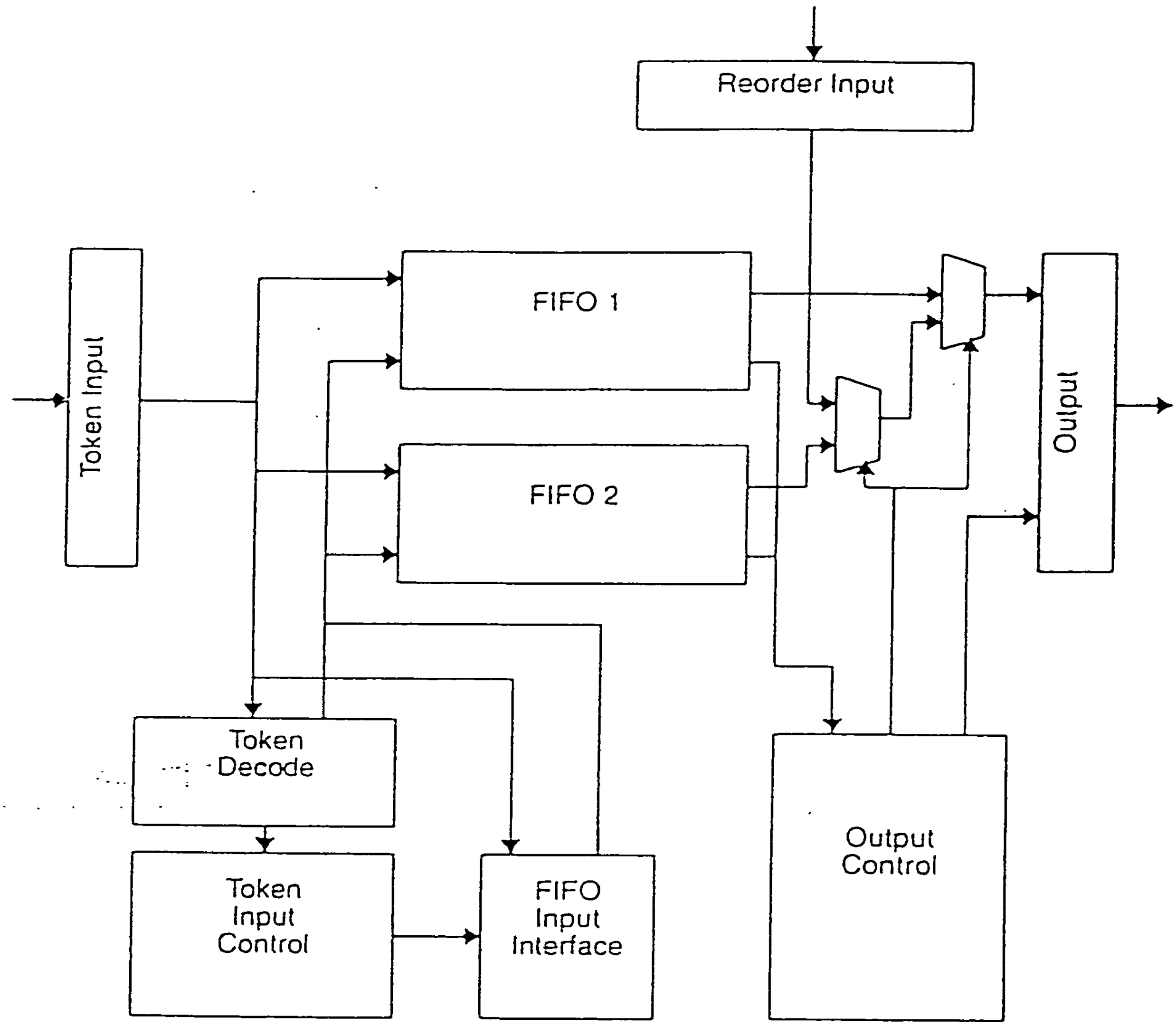

Reorder Input
Token Input
FIFO 1
FIFO 2
Output
Token Decode
Token Input Control
FIFO Input Interface
Output Control

FIG. 150

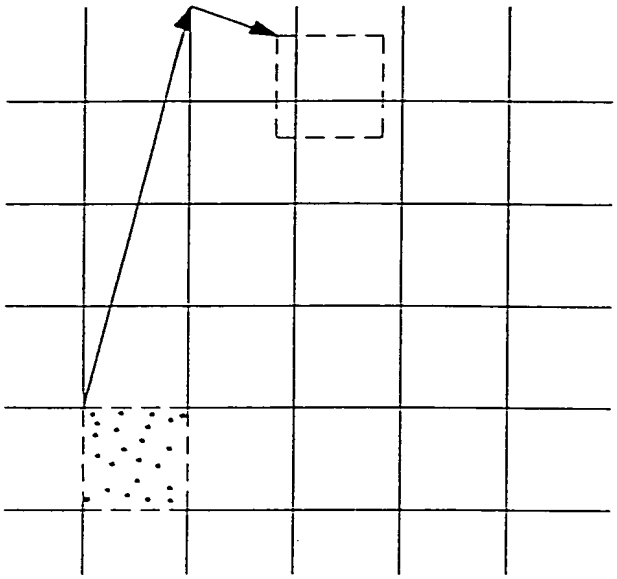


FIG. 151

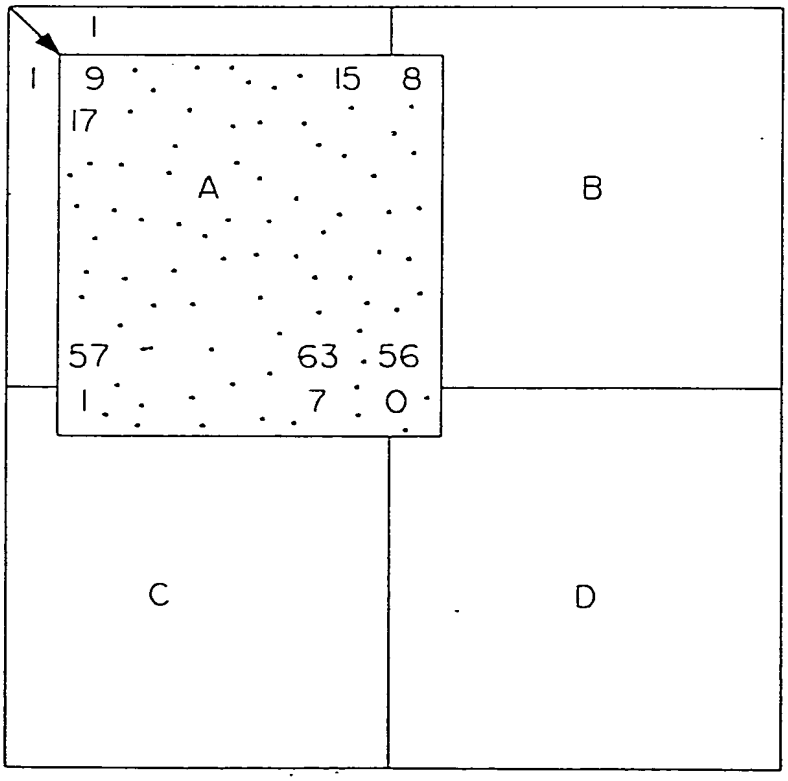


FIG. 152

Read Cycle

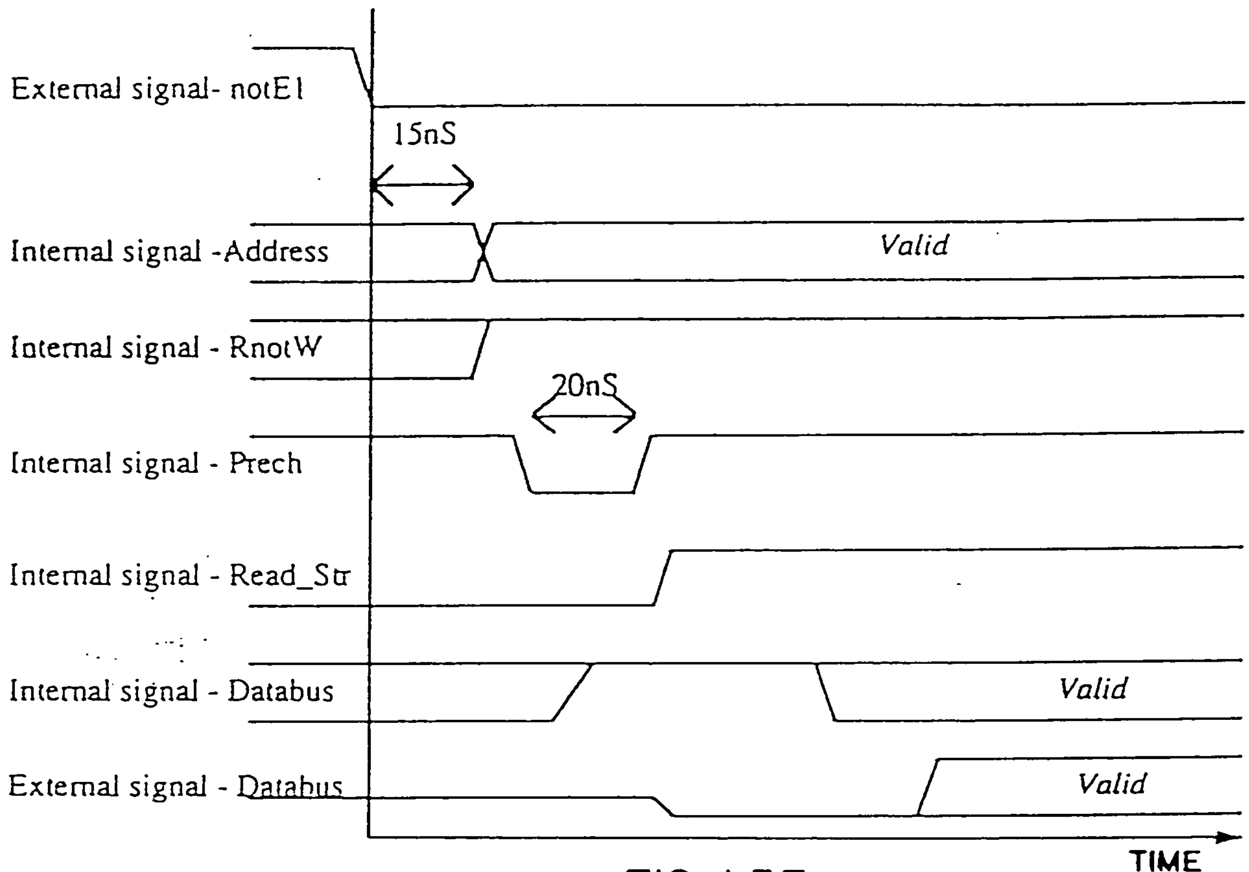


FIG. 153

Write Cycle

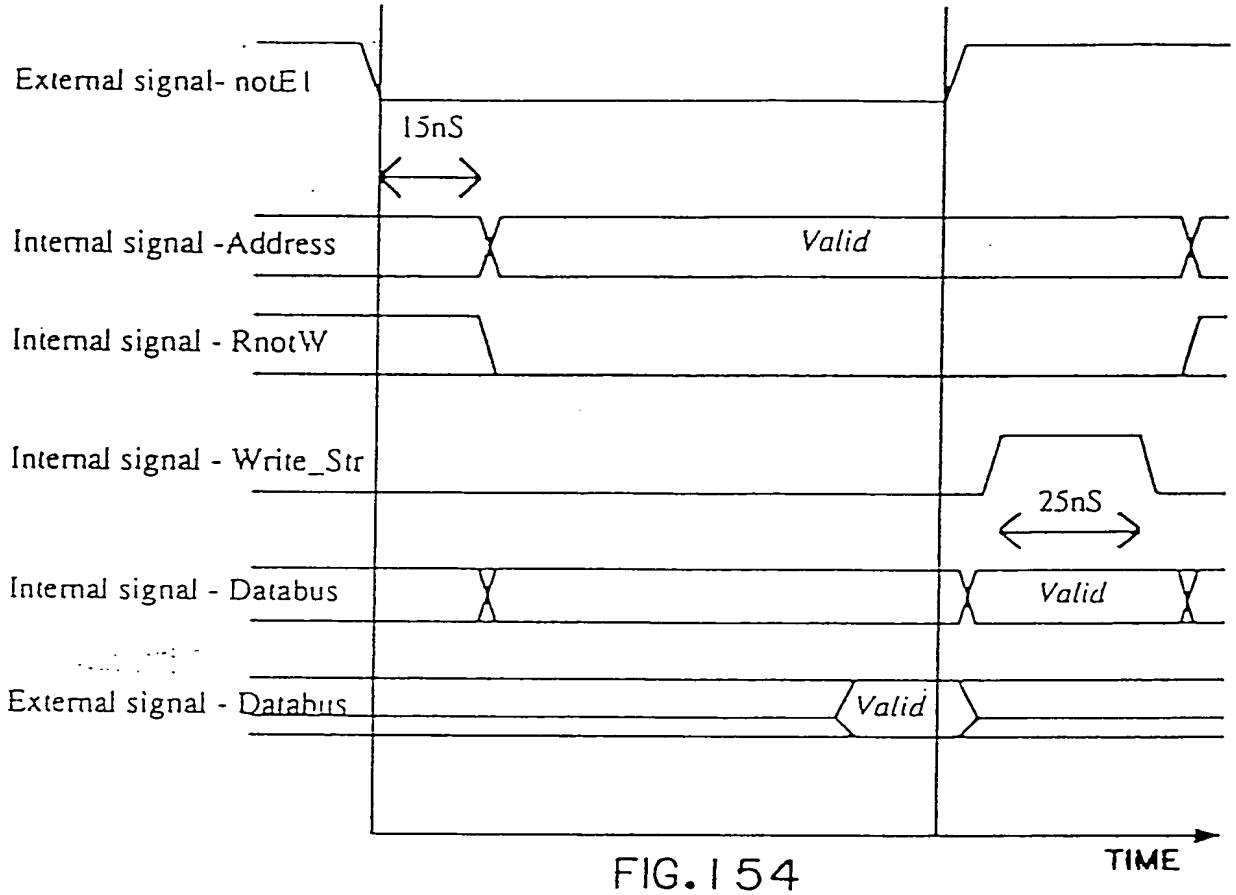


FIG. 154

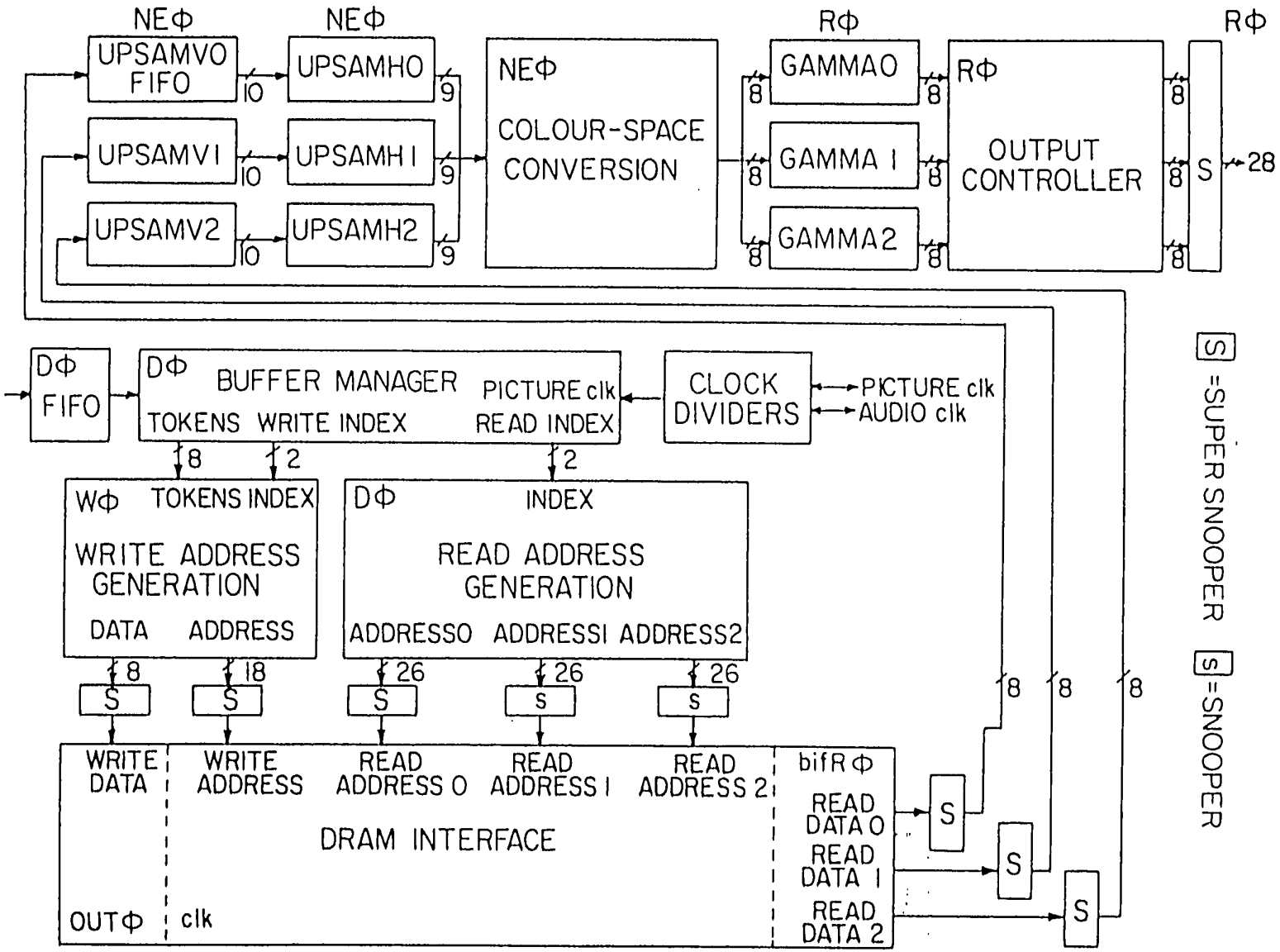


FIG. 155

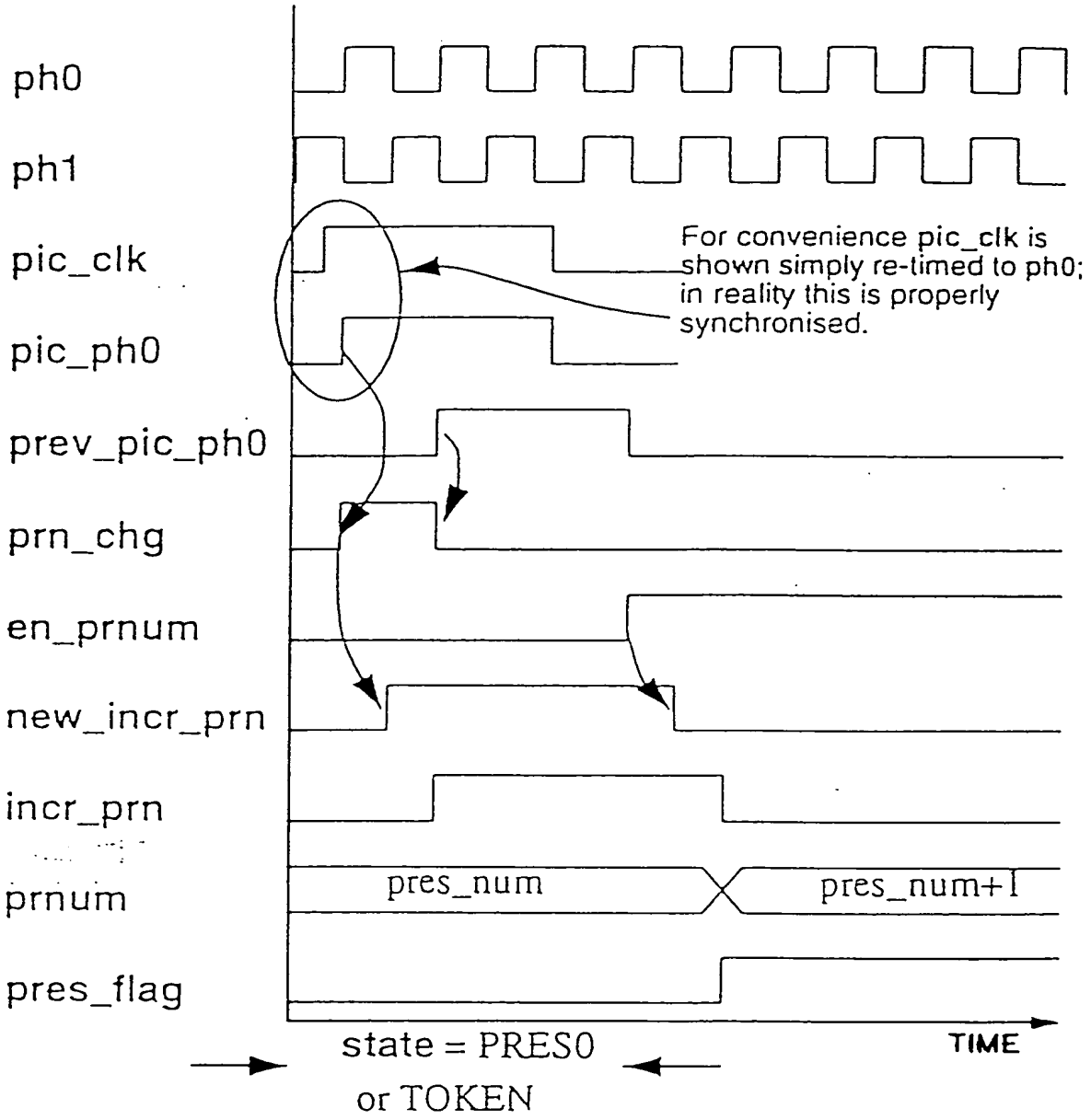

ph0
ph1
pic_clk
pic_ph0
prev_pic_ph0
prn_chg
en_prnum
new_incr_prn
incr_prn
prnum
pres_flag
pres_num
pres_num+1
For convenience pic_clk is shown simply re-timed to ph0; in reality this is properly synchronised.
state = PRES0
or TOKEN
TIME

FIG. 156

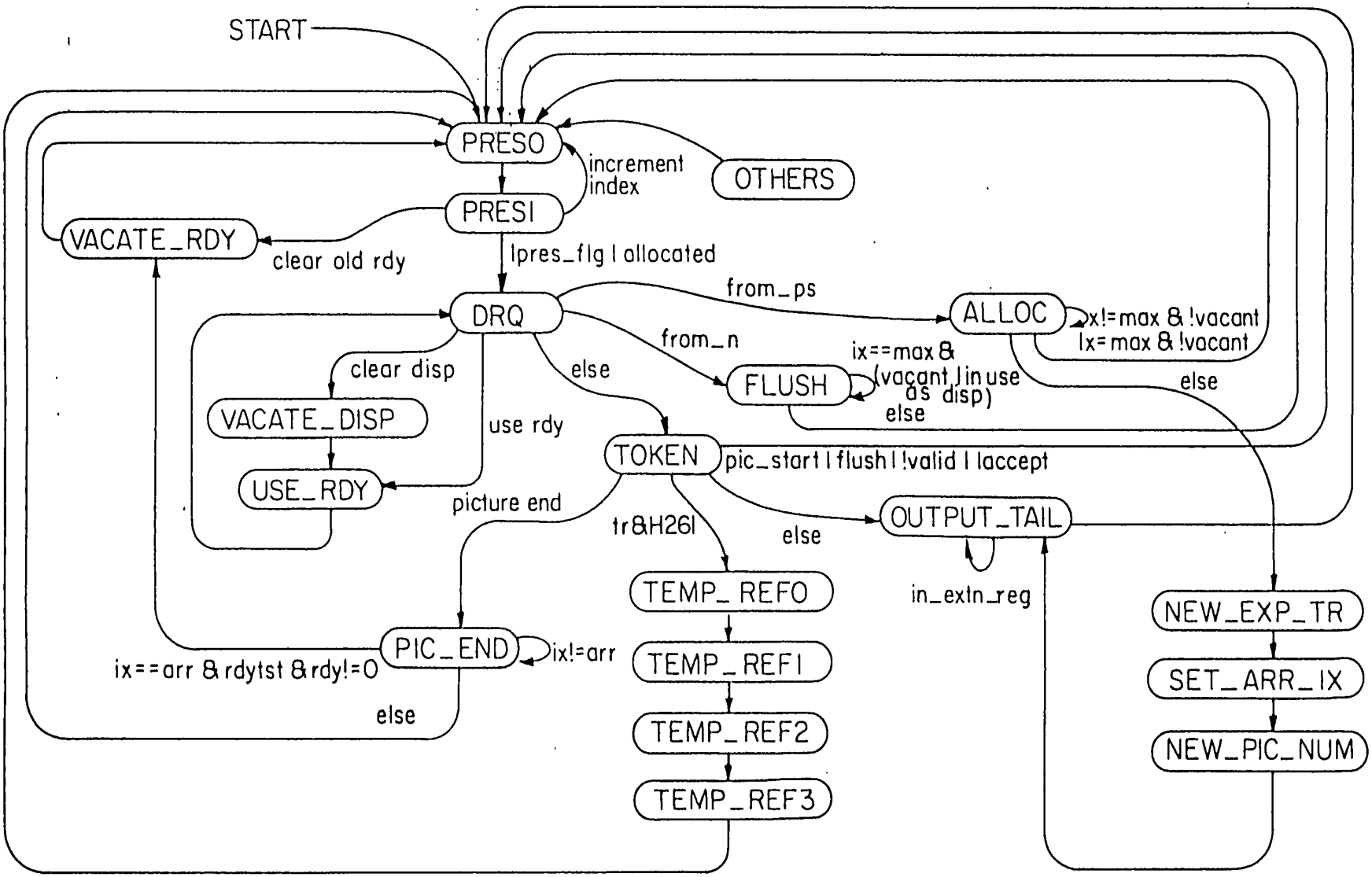

START
PRESO
increment index
OTHERS
PRESI
VACATE_RDY
clear old rdy
!pres_flg ! allocated
DRQ
from_ps
ALLOC
x!=max & !vacant
!x=max & !vacant
from_n
ix==max & (vacant ! in use as disp)
clear disp
else
FLUSH
else
else
VACATE_DISP
use rdy
TOKEN
pic_start ! flush ! !valid ! !accept
USE_RDY
picture end
tr8H26!
else
OUTPUT_TAIL
in_extn_reg
TEMP_REF0
NEW_EXP_TR
PIC_END
ix!=arr
ix==arr & rdytst & rdy!=0
TEMP_REF1
SET_ARR_IX
else
TEMP_REF2
NEW_PIC_NUM
TEMP_REF3

FIG. 157

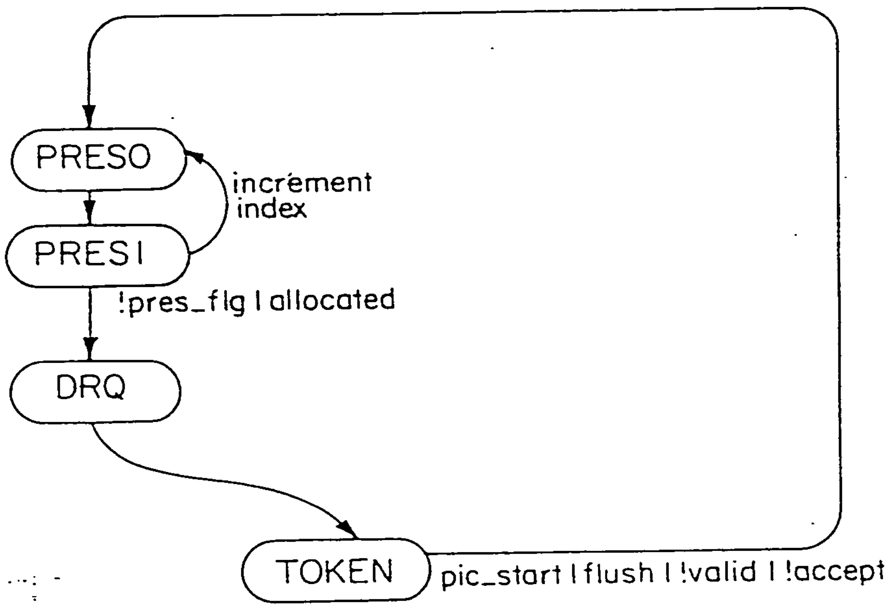


FIG. 158

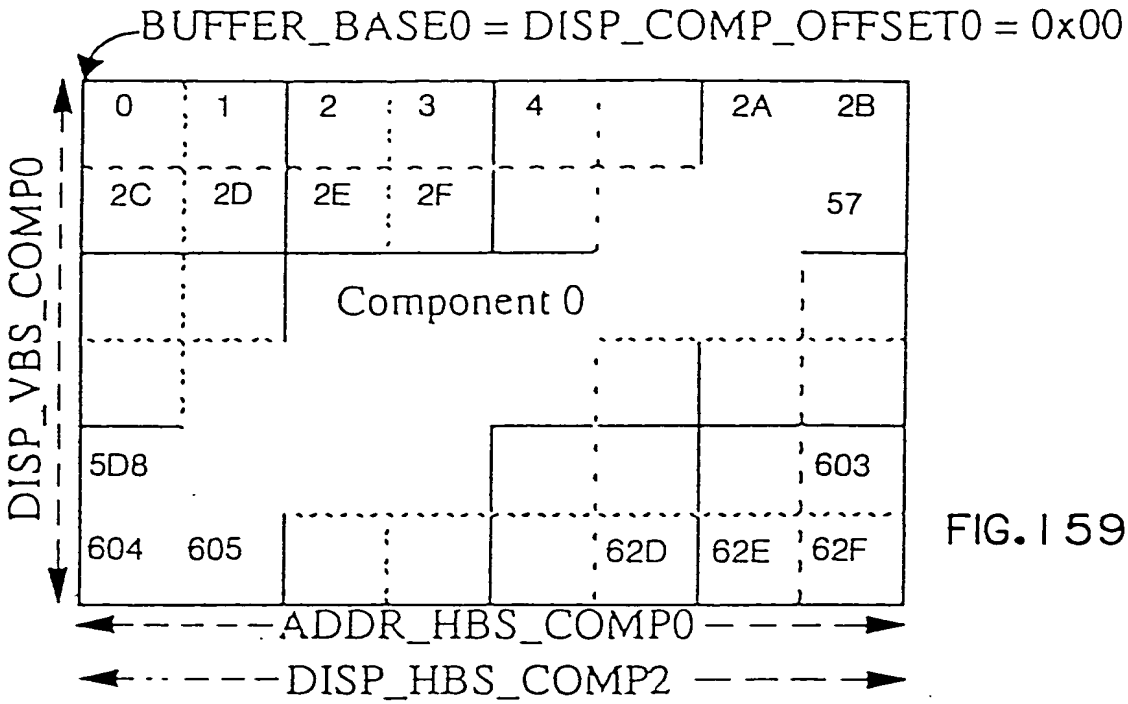


FIG. 159A

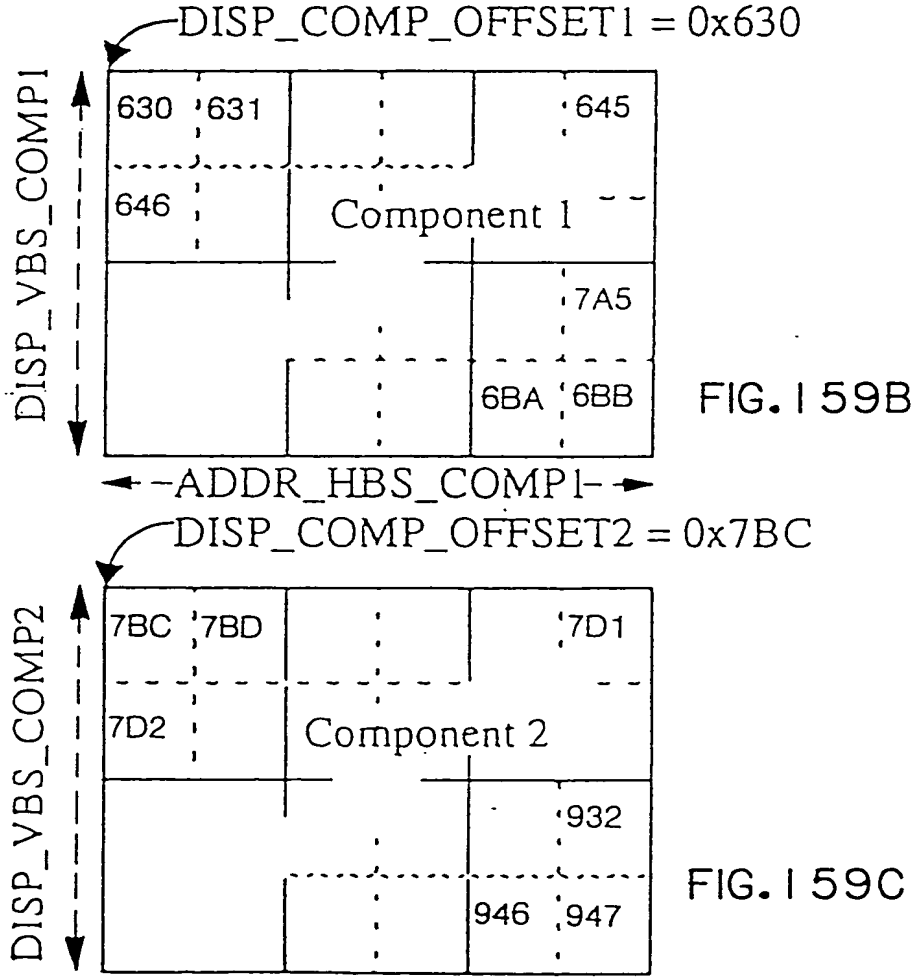
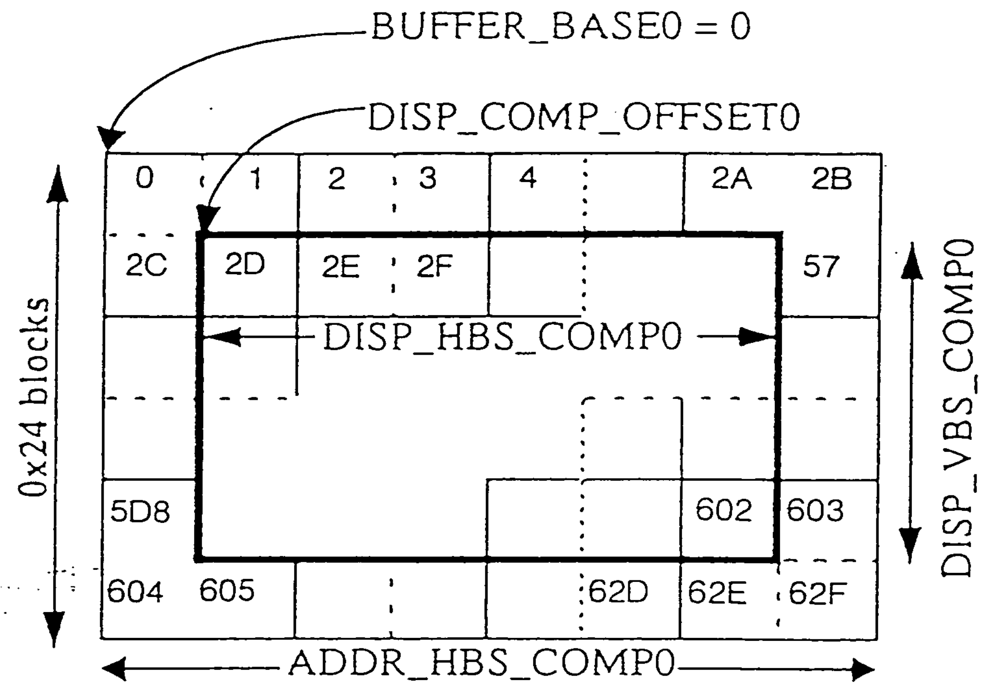


FIG. 159B

FIG. 159C



BUFFER_BASE0 = 0
DISP_COMP_OFFSET0
0
1
2
3
4
2A
2B
2C
2D
2E
2F
57
DISP_HBS_COMP0
5D8
602
603
604
605
62D
62E
62F
0x24 blocks
DISP_VBS_COMP0
ADDR_HBS_COMP0

FIG. 160

BUFFER OFFSET 0x00

COMPONENT OFFSET 0x000 + ...

00	01	02	03	04	05	06	07	08	09	0A	0B
0C	0D	0E	0F	10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F	20	21	22	23
24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B
3C	3D	3E	3F	40	41	42	43	44	45	46	47
48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B
6C	6D	6E	6F	70	71	72	73	74	75	76	77
78	79	7A	7B	7C	7D	7E	7F	80	81	82	83
84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

FIG. 161A

COMPONENT1 OFFSET 0x100 + ...

00	01	02	03	04	05
06	07	08	09	0A	0B
0C	0D	0E	0F	10	11
12	13	14	15	16	17
18	19	1A	1B	1C	1D
1E	1F	20	21	22	23

FIG. 161B

COMPONENT1 OFFSET 0x200 + ...

00	01	02	03	04	05
06	07	08	09	0A	0B
0C	0D	0E	0F	10	11
12	13	14	15	16	17
18	19	1A	1B	1C	1D
1E	1F	20	21	22	23

FIG. 161C

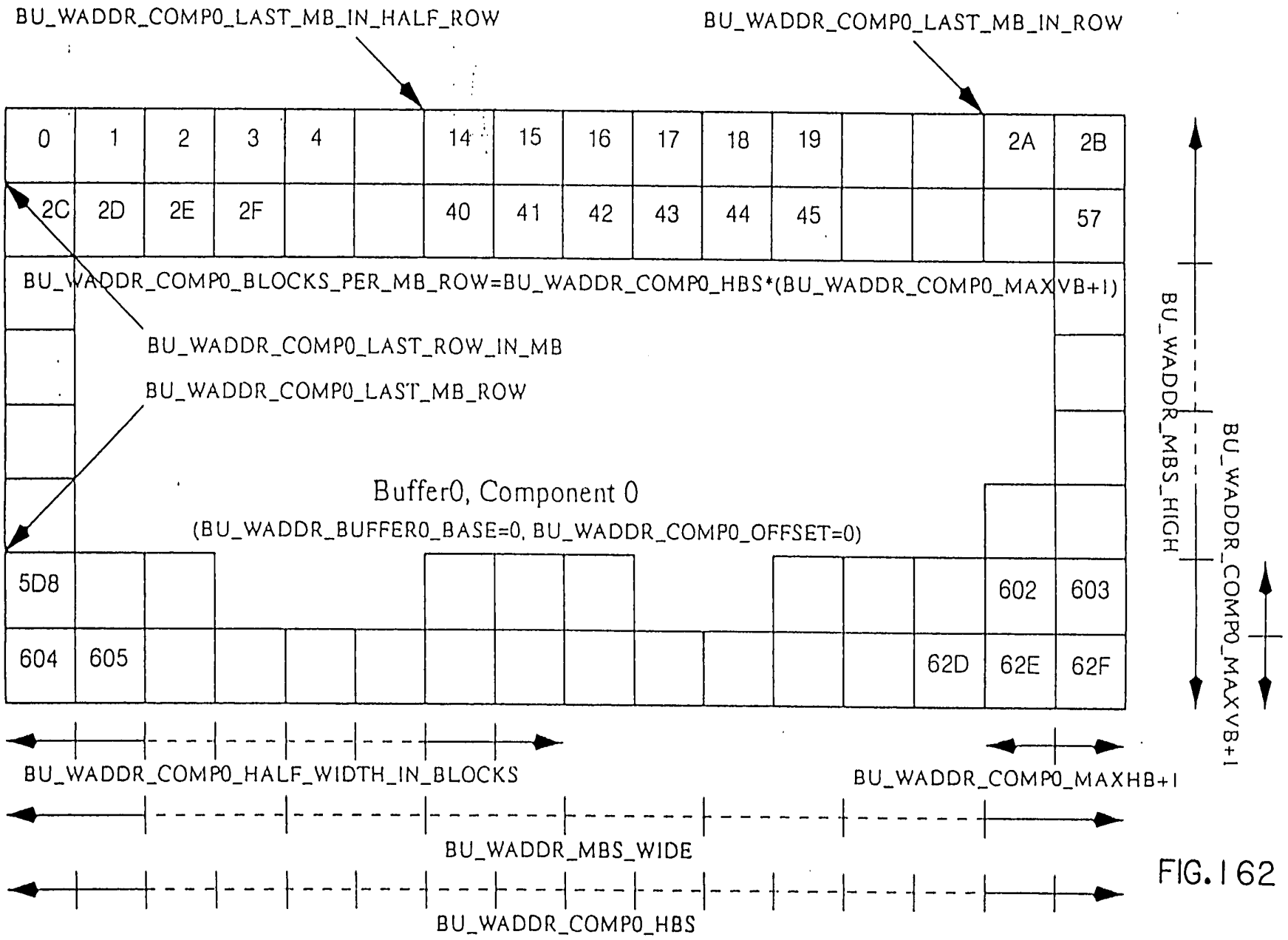
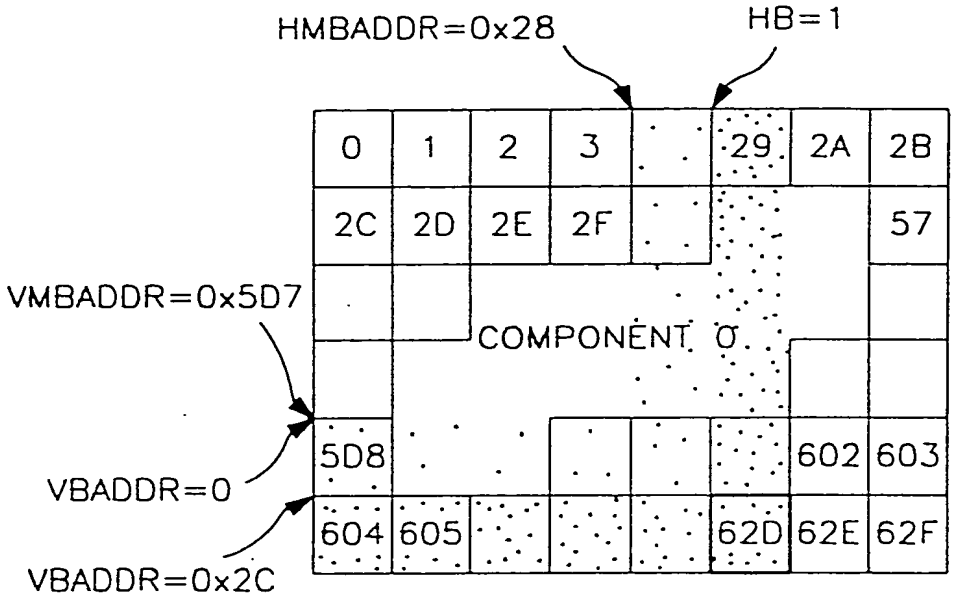
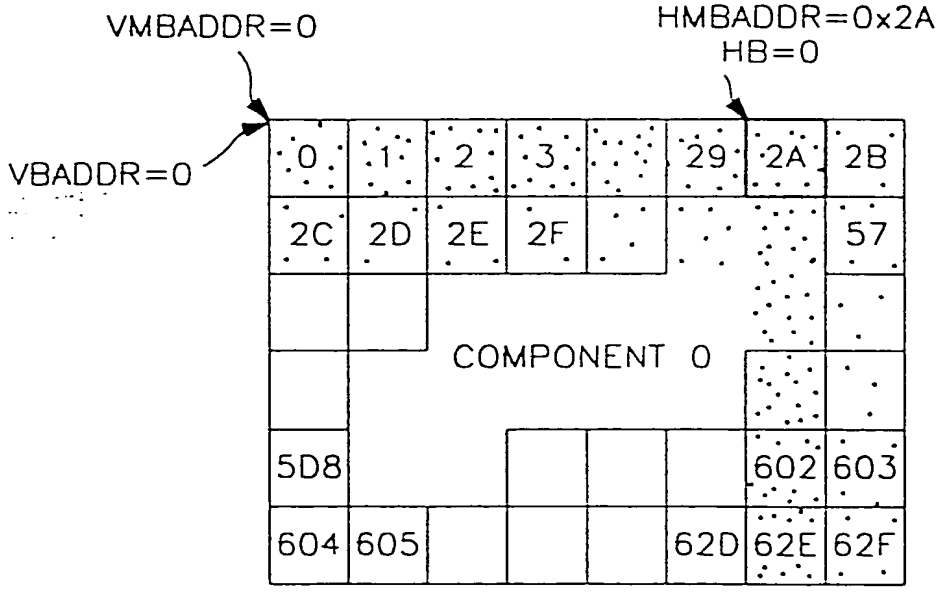

BU_WADDR_COMP0_LAST_MB_IN_HALF_ROW
BU_WADDR_COMP0_LAST_MB_IN_ROW
0 1 2 3 4 14 15 16 17 18 19 2A 2B
2C 2D 2E 2F 40 41 42 43 44 45 57
BU_WADDR_COMP0_BLOCKS_PER_MB_ROW=BU_WADDR_COMP0_HBS*(BU_WADDR_COMP0_MAXVB+1)
BU_WADDR_COMP0_LAST_ROW_IN_MB
BU_WADDR_COMP0_LAST_MB_ROW
Buffer0, Component 0
(BU_WADDR_BUFFER0_BASE=0, BU_WADDR_COMP0_OFFSET=0)
5D8 602 603
604 605 62D 62E 62F
BU_WADDR_MBS_HIGH
BU_WADDR_COMP0_MAXVB+1
BU_WADDR_COMP0_HALF_WIDTH_IN_BLOCKS
BU_WADDR_COMP0_MAXHB+1
BU_WADDR_MBS_WIDE
BU_WADDR_COMP0_HBS

FIG. 162



BLOCK ADDRESS=0+0+0x5D8+0x28+0x2C+1=0x62D

FIG. 163A



BLOCK ADDRESS=0+0+0+0x2A+0+0=0x2A

FIG. 163B

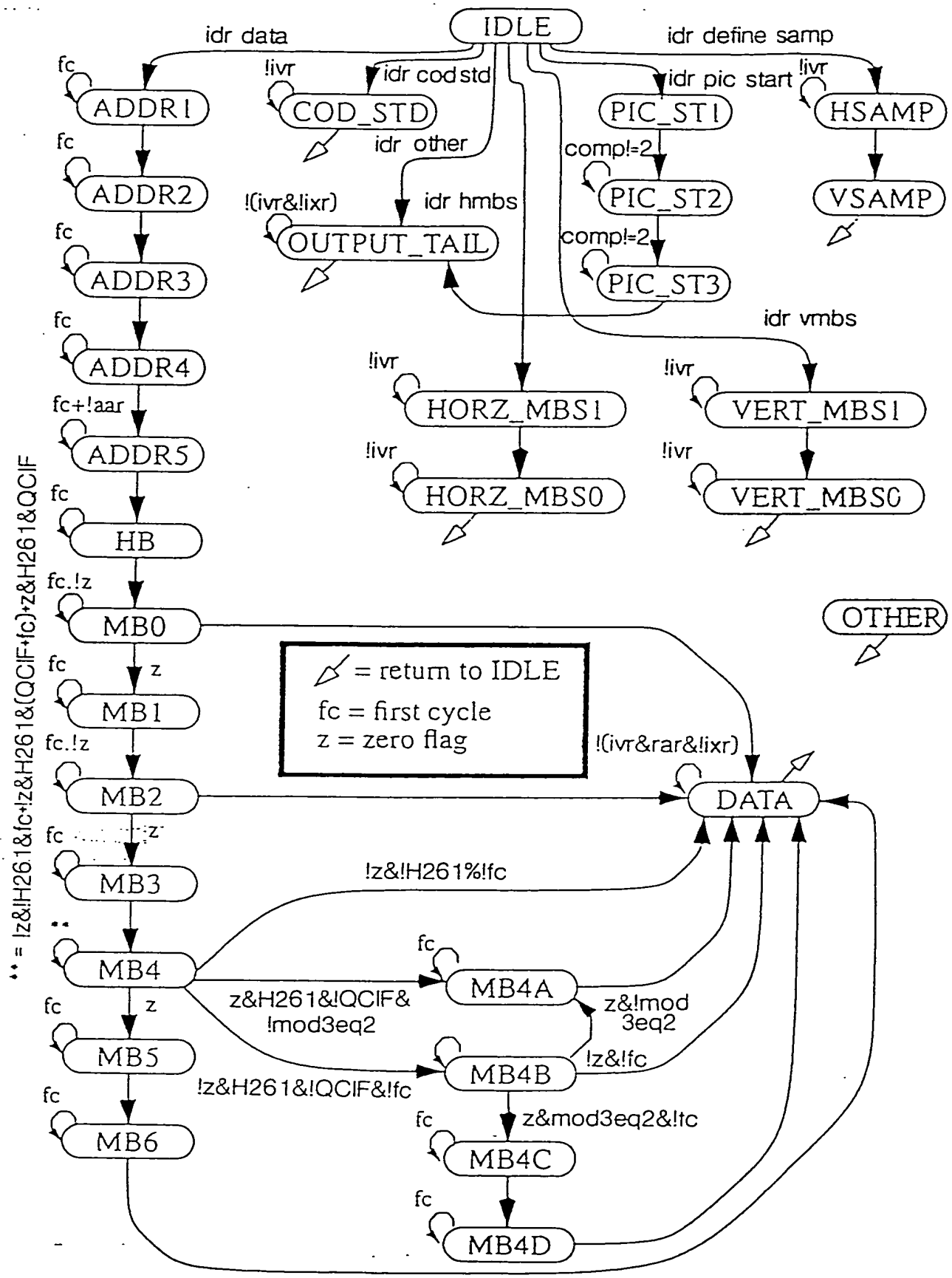

IDLE
idr data
idr define samp
idr cod std
idr other
idr pic start
idr hmbs
idr vmbs
!ivr
fc
ADDR1
ADDR2
ADDR3
ADDR4
ADDR5
HB
MB0
MB1
MB2
MB3
MB4
MB5
MB6
COD_STD
OUTPUT_TAIL
PIC_ST1
PIC_ST2
PIC_ST3
HSAMP
VSAMP
HORZ_MBS1
HORZ_MBS0
VERT_MBS1
VERT_MBS0
OTHER
DATA
MB4A
MB4B
MB4C
MB4D
comp!=2
!(ivr&!ixr)
fc+!aar
fc.!z
z
!(ivr&rar&!ixr)
!z&!H261%!fc
z&H261&!QCIF&
!mod3eq2
z&!mod
3eq2
!z&!fc
!z&H261&!QCIF&!fc
z&mod3eq2&!tc
= return to IDLE
fc = first cycle
z = zero flag
** = !z&!H261&!fc+!z&H261&(QCIF+fc)+z&H261&QCIF

FIG. 164

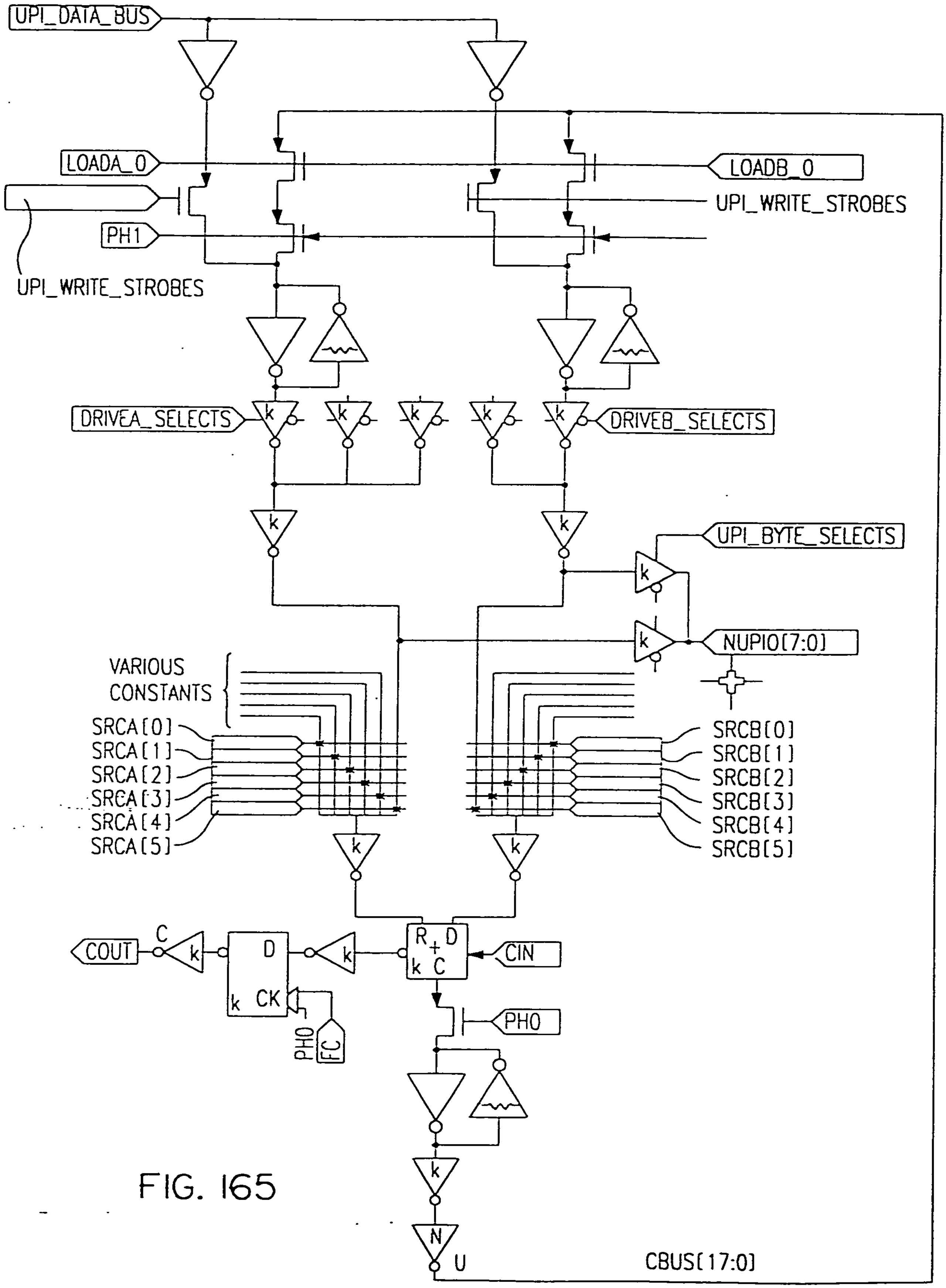


FIG. 165

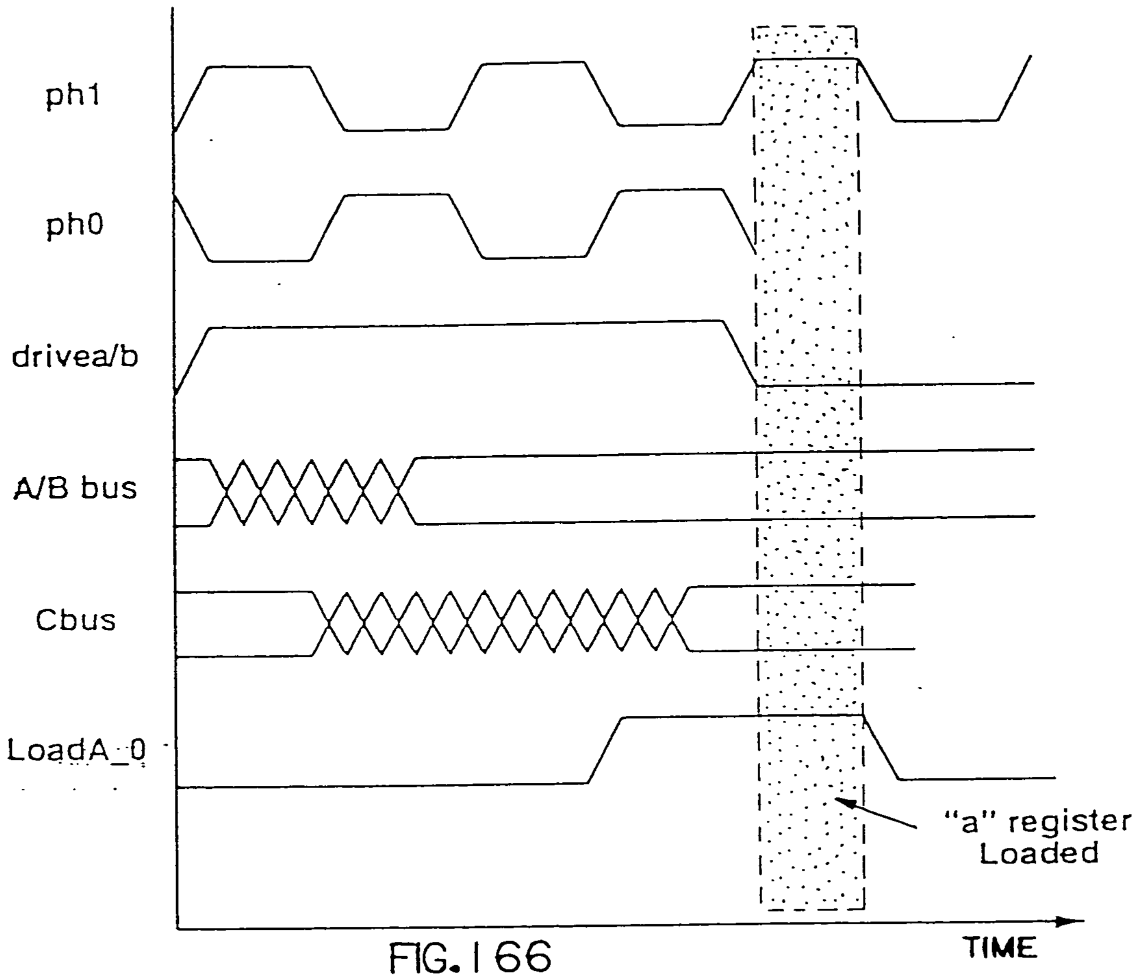


FIG. 166

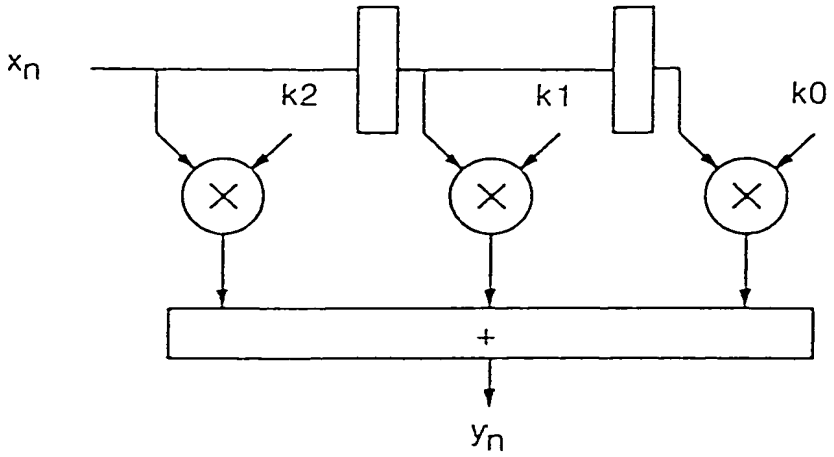


FIG. 167

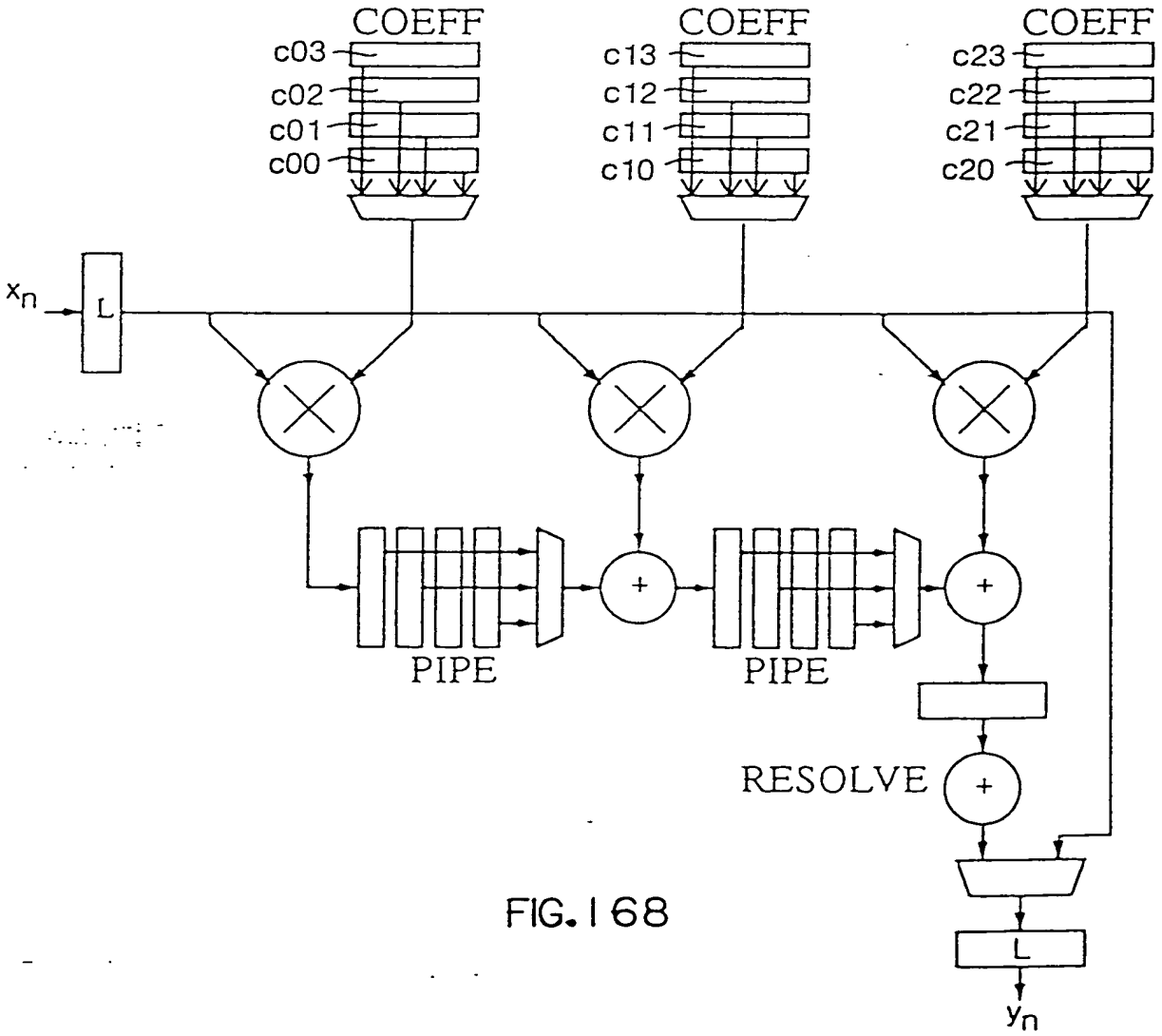


FIG. 168

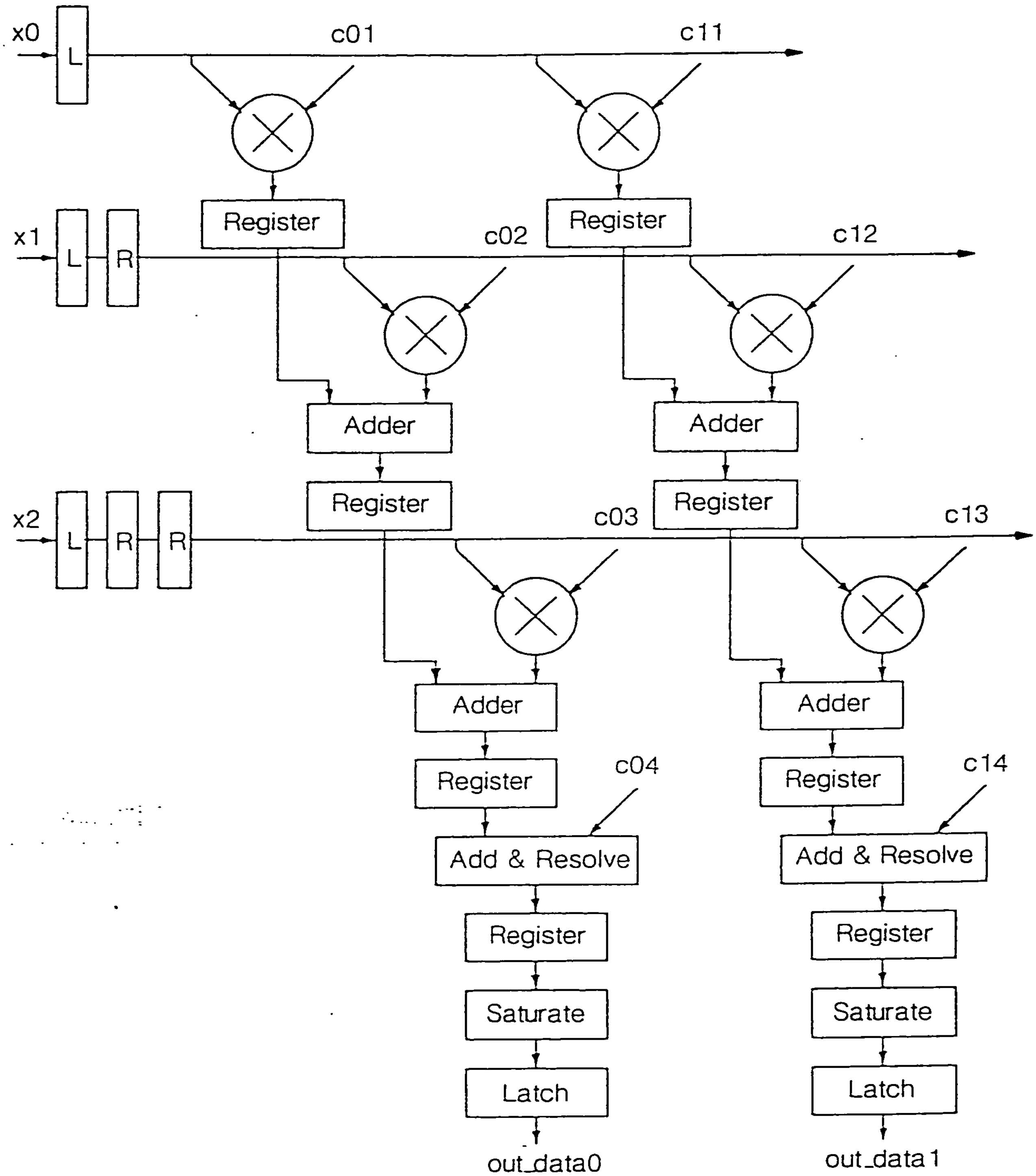

x0
L
c01
c11
Register
Register
x1
L
R
c02
c12
Adder
Adder
Register
Register
x2
L
R
R
c03
c13
Adder
Adder
Register
Register
c04
c14
Add & Resolve
Add & Resolve
Register
Register
Saturate
Saturate
Latch
Latch
out_data0
out_data1

FIG. 169